使用说明



78K0S/Kx1+

举例程序 (串行接口 UART6)

使用环形接收缓冲区进行全双工通信

本文档介绍了举例程序的操作概述，并介绍了如何运用举例程序以及如何设置和应用串行接口 UART6。本举例程序中，波特率设置为 9，600 bps，执行串行通信，并且跟据所接收的 1 个字符数据来发送 4 个字符数据。类似地，在收到错误数据的情况下，则根据该错误数据来发送相应 4 个字符数据。

目标器件：
78K0S/KA1+微控制器
78K0S/KB1+微控制器

目录

文档编号 U18914CA1V0AN00（第一版）
发布日期 2008 年 03 月 N

Windows 和 Windows XP 是微软在美国和/或其它国家的注册商标或商标。

M8E 02.11-1

第一章 概要

本举例程序提供了一个使用串行接口 UART6 进行全双工通信的应用实例。波特率设置为 9，600 bps，执行串行通信，并且根据所接收的 1 个字符数据发送 4 个字符数据。类似地，在收到错误数据的情况下，则根据该错误数据来发送相应 4 个字符数据。

1.1 初始设置的主要内容

初始设置的主要内容如下：

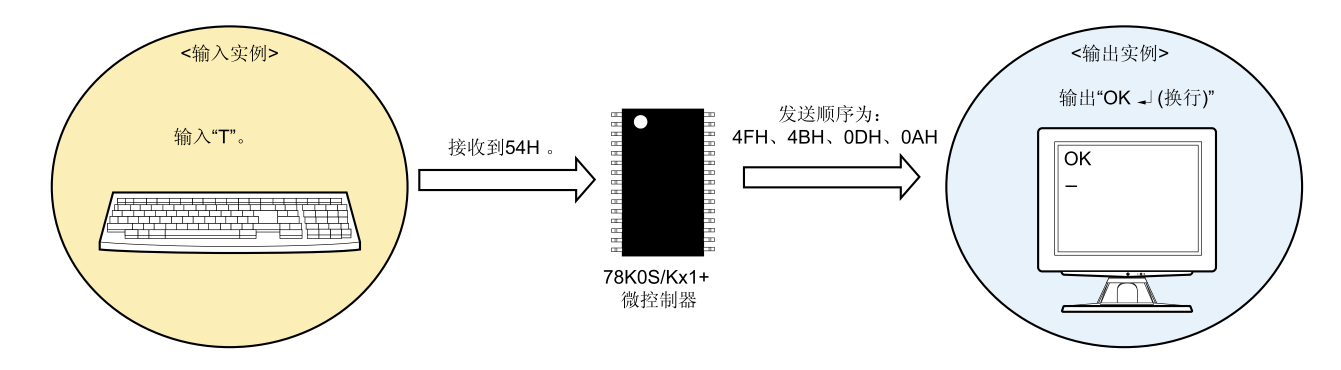
- 选择晶体振荡时钟或陶瓷振荡时钟作为系统时钟信号源[注]。
- 停止看门狗定时器的运行。
- 将 V_{LVI}（低电压检测电压）设置为 4.3 V ±0.2 V。
- V_{DD}（电源电压）高于或等于 V_{LVI} 后，当检测到 V_{DD} 低于 V_{LVI} 时，产生内部复位（LVI 复位）信号。
- 将 CPU 时钟频率设置为 8 MHz。
- 设置 I/O 端口。
- 设置串行接口 UART6：
  - 波特率：9，600 bps。
  - 数据字符长度： 7 位。
  - 奇偶校验说明：偶校验。
  - 停止位位数：1 位。
  - 起始位说明：LSB 在先传输。
  - TxD6 输出：正常输出。
  - 产生 INTSRE6 作为错误发生时的中断。
  - 内部工作时钟操作使能。
  - 发送操作使能。
  - 接收操作使能。

注　　使用选项字节进行此项设置。

[专栏] 什么是全双工通信？
全双工通信是一种能够同时独立执行发送操作和接收操作的通信方式。串行接口 UART6 支持全双工通信，允许同时发送和接收数据。

1.2　主循环的后续内容

初始设置完成后，由 RxD6 引脚输入的数据启动串行通信的接收操作。本举例程序中，假定接收和发送的数据为 ASCII 码形式，并且根据所接收的 1 个字符数据发送 4 个字符数据。类似地，在收到错误数据的情况下，则根据该错误数据来发送相应 4 个字符数据。



在 RAM 区域中，受保护的一个 50 字节（= 1 字节（1 个数据）× 50）存储区用作存储接收数据的缓冲器。

执行中断服务时，因为接收数据已存储在缓冲器中且从缓冲区的起始地址处连续存储，所以，可以按顺序存储接收数据。而且，缓冲器已设定为一个环形缓冲区，当接收数据存储到缓冲区的末尾时，接收数据又从缓冲区的起始地址处开始存储。当缓冲器中有空闲存储空间时，接收数据则存储在缓冲器内，但是，当没有空闲存储空间时，接收数据就被丢弃而不存储。

注意事项　关于使用该器件时的注意事项，参见各产品（78K0S/KA1+、78K0S/KB1+）的用户手册。


[专栏] 什么是 ASCII 码？

ASCII 码是一种使用 7 位来表示一个字符（字母、数字、符号、控制字符）的编码形式。


[专栏] 什么是环形缓冲区？

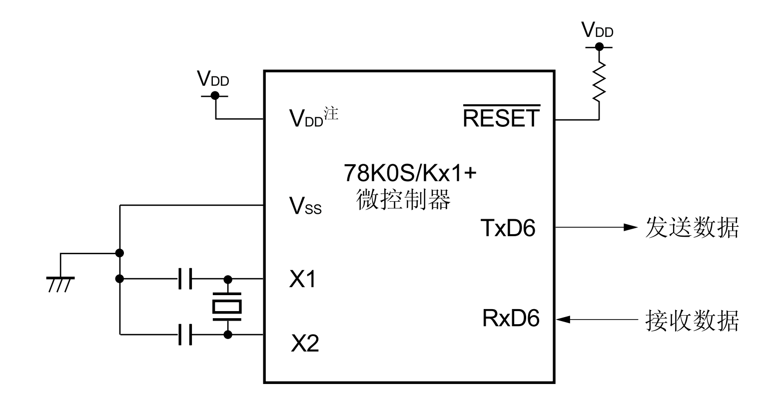
环形缓冲区是一种控制发送或接收缓冲器的方式。环形缓冲区是一块预先分配好的固定缓冲区域，按照传送至缓冲区的数据顺序处理数据，并控制缓冲器将缓冲区末尾的后续地址变为起始地址。该缓冲区可循环存储数据，故此命名为环形缓冲区。

第二章　电路图

本章介绍了该举例程序中所用的电路图。

2.1　电路图

电路图如下所示：



注　　适用电压范围为 4.5 V ≤ V_{DD} ≤ 5.5 V。

注意事项 **1.** 将 **AV_{REF}** 引脚直接连接至 **V_{DD}**。

2. 将 **AV_{SS}** 引脚直接连接至 **GND**（仅适用于 **78K0S/KB1+**微控制器）。

3. 除电路图中所示引脚和 **AV_{REF}** 引脚、**AV_{SS}** 引脚之外，保留所有未用引脚为开路状态（未连接）。

第三章　软件

本章介绍了所下载压缩文件的组成、所用微控制器的内部外设功能、以及初始设置和本举例程序的操作概要，并提供了流程图。

3.1　文件的组成

下表显示了所下载压缩文件的组成：

文件名称	说明	包含的压缩(*.zip)文件	
main.asm (汇编语言版) main.c (C语言版)	用于硬件初始化处理的源文件和微控制器的主处理程序。	●注	●注
op.asm	用于设置选项字节的汇编语言源文件（设置系统时钟信号源）。	●	●
uart6.prw	用于集成开发环境PM+的工作空间文件。		●
uart6.prj	用于集成开发环境PM+的工程文件。		●

注　汇编语言版本包含“main.asm”文件，C 语言版本包含“main.c”文件。

备注


：　仅包含源文件。


：　包含用于集成开发环境 PM+的文件。

3.2　所使用的内部外设功能

本举例程序中使用了微控制器的下列内部外设功能：

- 串行通信：　　　　　串行接口 UART6。
- $V_{DD} < V_{LVI}$ 检测：　　低电压检测器(LVI)。
- 数据的输入输出：　　RxD6 和 TxD6。

3.3　初始设置和操作概要

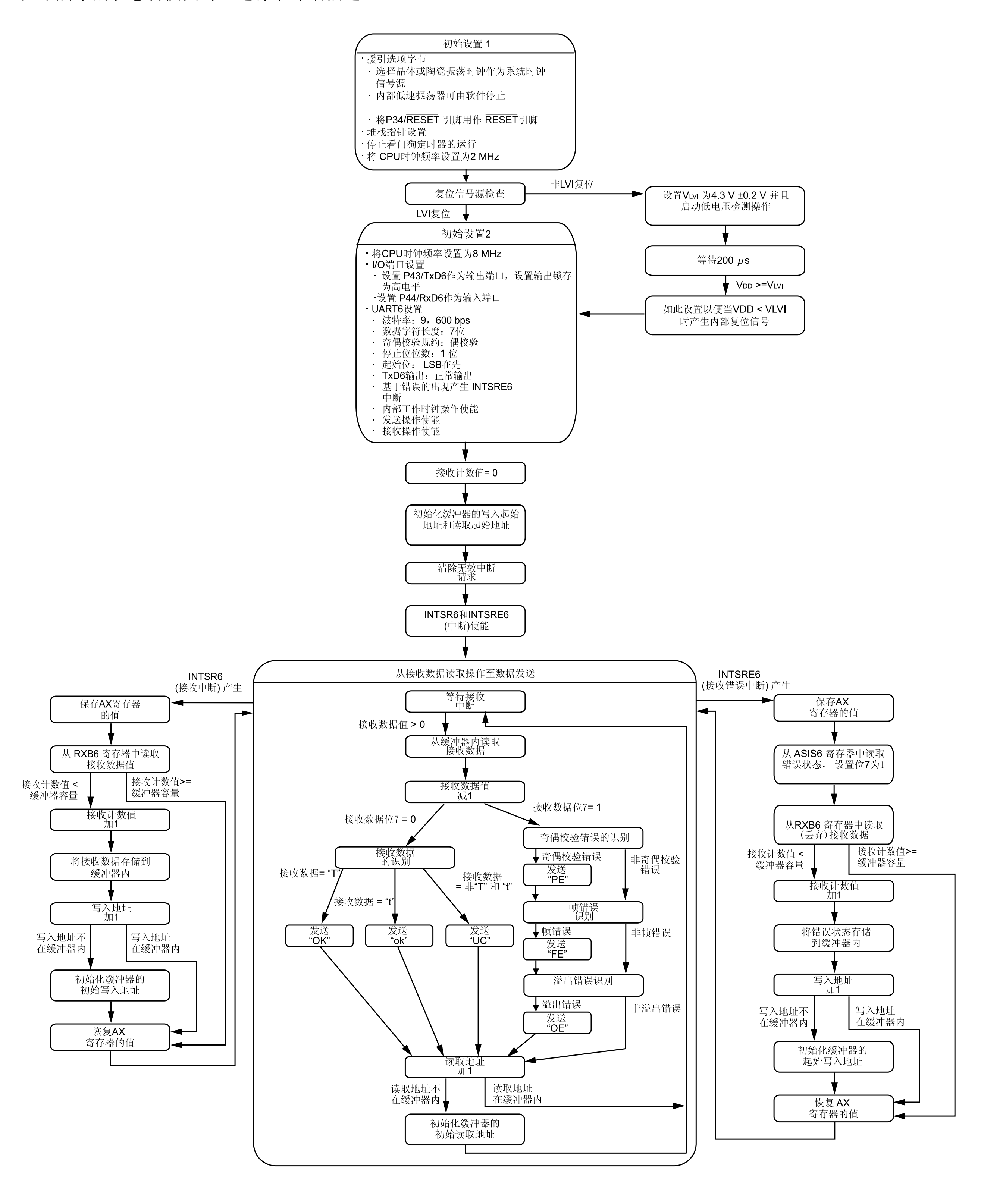
本举例程序中，初始设置时完成低电压检测功能的设置、时钟频率的选择、I/O 端口的设置以及串行接口 UART6 的设置。

初始设置完成后，由 RxD6 引脚输入的数据启动串行通信的接收操作。本举例程序中，假定接收和发送的数据为 ASCII 码形式，并且根据所接收的 1 个字符数据发送 4 个字符数据。类似地，在收到错误数据的情况下，则根据该错误数据来发送相应 4 个字符数据。

在 RAM 区域中，受保护的一个 50 字节（= 1 字节（1 个数据）× 50）的存储区用作存储接收数据的缓冲器。

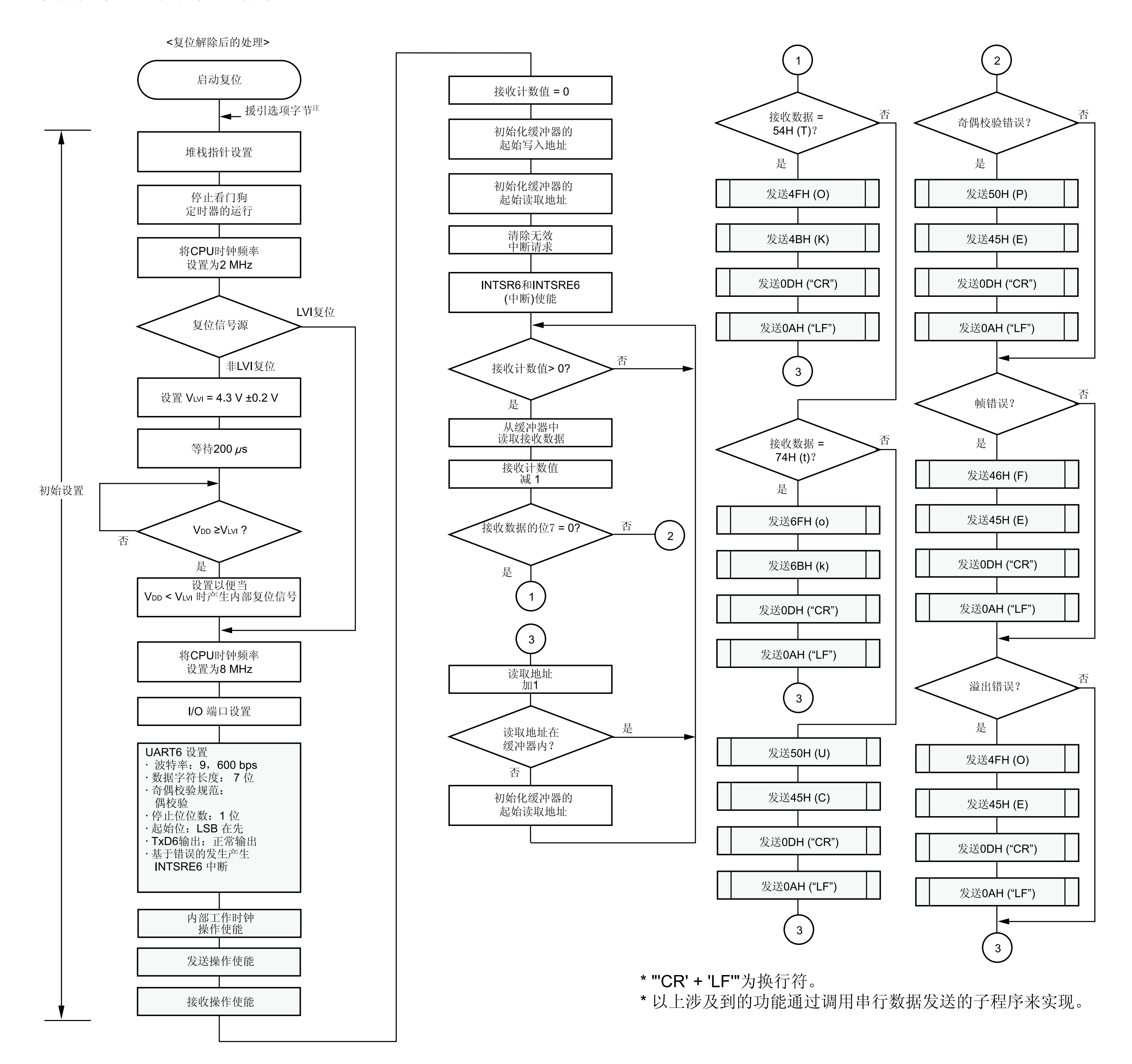
执行中断服务时，因为接收数据已存储在缓冲器中且从缓冲区的起始地址处连续存储，所以，可以按顺序接收数据。而且，缓冲器已设定为一个环形缓冲区，接收数据存储到缓冲区的末尾时，接收数据又从缓冲区的起始地址处开始存储。当缓冲器中有空闲存储空间时，接收数据存储在缓冲器内，但是，当没有空闲存储空间时，接收数据就被丢弃而不存储。

如下所示的状态转换图对此进行了详细描述：



3.4　流程图

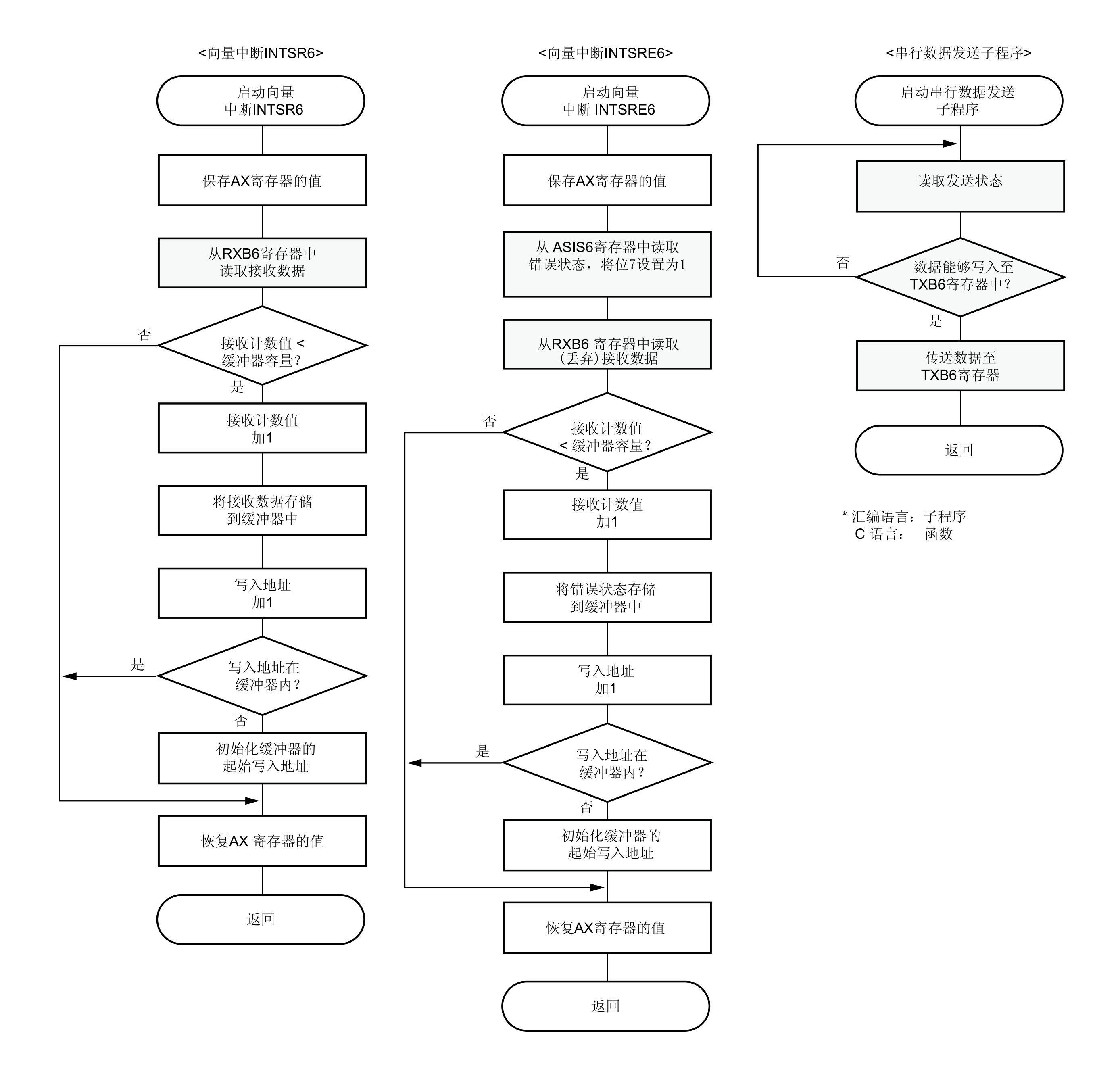
举例程序的流程图如下所示：



注　　复位解除后，微控制器自动援引选项字节。本举例程序中，援引选项字节设置下列内容：

- 使用晶体振荡时钟或陶瓷振荡时钟作为系统时钟信号源。
- 内部低速振荡器可由软件停止。
- 将 P34/$\overline{\text{RESET}}$ 引脚用作 $\overline{\text{RESET}}$ 引脚。

备注　　<向量中断 INTSR6>、<向量中断 INTSRE6>、及<串行数据发送子程序>的流程图如下页所示：


<向量中断INTSR6>
启动向量
中断INTSR6
保存AX寄存器的值
从RXB6寄存器中
读取接收数据
接收计数值 <
缓冲器容量？
否
是
接收计数值
加1
将接收数据存储
到缓冲器中
写入地址
加1
写入地址在
缓冲器内？
是
否
初始化缓冲器的
起始写入地址
恢复AX 寄存器的值
返回
<向量中断INTSRE6>
启动向量
中断 INTSRE6
保存AX寄存器的值
从 ASIS6寄存器中读取
错误状态，将位7设置为1
从RXB6 寄存器中读取
（丢弃）接收数据
接收计数值
< 缓冲器容量？
否
是
接收计数值
加1
将错误状态存储
到缓冲器中
写入地址
加1
写入地址在
缓冲器内？
是
否
初始化缓冲器的
起始写入地址
恢复AX寄存器的值
返回
<串行数据发送子程序>
启动串行数据发送
子程序
读取发送状态
数据能够写入至
TXB6寄存器中？
否
是
传送数据至
TXB6寄存器
返回
* 汇编语言：子程序
C 语言： 函数

第四章　设置方法

本章介绍了串行接口 UART6 的设置。

关于其它初始设置，参见 **78K0S/Kx1+举例程序(初始设置)LED 灯开关控制的使用说明**。关于中断，参见 **78K0S/Kx1+举例程序(中断)由开关输入产生外部中断的使用说明**。关于低电压检测(LVI)，参见 **78K0S/Kx1+举例程(低电压检测)电压低于 2.7 V 的复位使用说明**。

关于如何设置寄存器，参见各产品（**78K0S/KA1+**、**78K0S/KB1+**）的用户手册。

关于汇编语言指令，参见 **78K/0S 系列指令用户手册**。

4.1　设置串行接口UART6

串行接口 UART6 使用下列 11 种寄存器：

- 时钟选择寄存器 6 (CKSR6)。
- 波特率发生器控制寄存器 6 (BRGC6)。
- 异步串行接口工作模式寄存器 6 (ASIM6)。
- 异步串行接口控制寄存器 6 (ASICL6)。
- 发送缓冲器寄存器 6 (TXB6)。
- 接收缓冲器寄存器 6 (RXB6)。
- 异步串行接口发送状态寄存器 6 (ASIF6)。
- 异步串行接口接收错误状态寄存器 6 (ASIS6)。
- 输入开关控制寄存器 (ISC)。
- 端口模式寄存器 x (PMx)[注1]。
- 端口寄存器 x (Px)[注1]。

注　**1.** 所用串行接口 UART6 的引脚设置如下：

POWER6	TXE6	RXE6	PM43	P43	PM44	P44	UART6 工作模式	引脚功能	
								TxD6/INTP1/P43	RxD6/P44
0	0	0	×[注2]	×[注2]	×[注2]	×[注2]	停止	P43	P44
1	0	1	×[注2]	×[注2]	1	×	接收	P43	RxD6
	1	0	0	1	×[注2]	×[注2]	发送	TxD6	P44
	1	1	0	1	1	×	发送和接收	TxD6	RxD6

2. 用于设置端口功能。

备注　×：　不必理会。
POWER6：　ASIM6 寄存器的位 7。
TXE6：　ASIM6 寄存器的位 6。
RXE6：　ASIM6 寄存器的位 5。

<串行接口 UART6 的基本操作设置步骤实例>

<1>　利用 CKSR6 寄存器设置 UART6 的基准时钟(f_{XCLK6})。 ⎫ 波特率设置
<2>　利用 BRGC6 寄存器设置 UART6 的基准时钟(f_{XCLK6})分频系数。⎭
<3>　利用 ASIM6 寄存器设置奇偶校验、字符长度、停止位、以及错误中断。
<4>　利用 ASICL6 寄存器设置起始位与允许或禁止 TxD6 输出反转。
<5>　设置(1) POWER6：内部工作时钟操作使能。
<6>　设置(1) TXE6：发送操作使能。
<7>　设置(1) RXE6：接收操作使能。

注意事项 1. **步骤<6>之后，若要启动发送操作，至少要等待一个 UART6 基准时钟(f_{XCLK6})，然后才写入发送数据至 TXB6 寄存器中。**

2. **步骤<7>之后，经过一个 UART6 基准时钟(f_{XCLK6})之后，进入接收使能状态。**

3. **考虑到与其它通信方的关系，设置 PMx 寄存器和 Px 寄存器。使用 TxD6 引脚时，为了避免非法起始位（下降沿信号）的产生，将 P43 设置为 1 之后再将 PM43 设置为 0（输出）。**

(1) CKSR6 寄存器的设置

该寄存器用来选择串行接口 UART6 的基准时钟(f_{XCLK6})。

图 4-1. 时钟选择寄存器 6 (CKSR6)的格式

CKSR6

0	0	0	0	TPS63	TPS62	TPS61	TPS60

基准时钟(f_{XCLK6})的选择

TPS63	TPS62	TPS61	TPS60	
0	0	0	0	f_{XP}
0	0	0	1	$f_{XP}/2$
0	0	1	0	$f_{XP}/2^2$
0	0	1	1	$f_{XP}/2^3$
0	1	0	0	$f_{XP}/2^4$
0	1	0	1	$f_{XP}/2^5$
0	1	1	0	$f_{XP}/2^6$
0	1	1	1	$f_{XP}/2^7$
1	0	0	0	$f_{XP}/2^8$
1	0	0	1	$f_{XP}/2^9$
1	0	1	0	$f_{XP}/2^{10}$
1	0	1	1	$f_{XP}/2^{11}$
非上述设置				设置禁止

注意事项　将 POWER6 设置为 0 之后，再对 TPS63 至 TPS60 进行重写。

备注　**1.** 通信执行期间（POWER6 = 1 和 TXE6 = 1，或 POWER6 = 1 和 RXE6 = 1），可通过软件来刷新（写入相同值）CKSR6 寄存器的值。

2. f_{XP}：提供至外围硬件的时钟振荡频率。

(2) BRGC6 寄存器的设置

该寄存器用来选择串行接口 UART6 的基准时钟(f_{XCLK6})分频系数。

图 4-2. 波特率发生器控制寄存器 6 (BRGC6)的格式

BRGC6

MLD67	MLD66	MLD65	MLD64	MLD63	MLD62	MLD61	MLD60	k	8位计数器输出时钟的选择
0	0	0	0	0	x	x	x	x	设置禁止
0	0	0	0	1	0	0	0	8	f_{XCLK6}/8
0	0	0	0	1	0	0	1	9	f_{XCLK6}/9
0	0	0	0	1	0	1	0	10	f_{XCLK6}/10
•	•	•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•	•	•
1	1	1	1	1	1	0	0	252	f_{XCLK6}/252
1	1	1	1	1	1	0	1	253	f_{XCLK6}/253
1	1	1	1	1	1	1	0	254	f_{XCLK6}/254
1	1	1	1	1	1	1	1	255	f_{XCLK6}/255

注意事项 **1.** 将 **TXE6** 和 **RXE6** 都设为 **0** 之后，再对 **MLD67** 至 **MLD60** 进行重写。

2. 波特率值通过 **8** 位计数器输出时钟进行 **2** 分频得到：

• 波特率 = $\frac{f_{XCLK6}}{2 \times k}$ [bps]

备注 **1.** 通信执行期间（POWER6 = 1 和 TXE6 = 1，或 POWER6 = 1 和 RXE6 = 1），可通过软件来刷新（写入相同值）BRGC6 寄存器的值。

2. f_{XCLK6}：利用 TPS63 至 TPS60 来选择基准时钟的频率。

3. k：　利用 MLD67 至 MLD60 来设置该值（k = 8，9，10，...，255）。

4. ×：　不必理会。

(3) ASIM6 寄存器的设置

该寄存器用于控制串行接口 UART6 的串行通信工作。

图 4-3. 异步串行接口工作模式寄存器 6 (ASIM6)的格式

ASIM6

POWER6	TXE6[注1]	RXE6[注2]	PS61	PS60	CL6	SL6	ISRM6

错误发生时的中断说明

ISRM6	
0	INTSRE6
1	INTSR6 (与接收完成时产生的中断相同)。

发送数据的停止位位数说明

SL6	
0	停止位位数 = 1
1	停止位位数 = 2

发送和接收数据的字符长度说明

CL6	
0	数据字符长度 = 7 位
1	数据字符长度 = 8 位

奇偶校验位说明

PS61	PS60	发送操作	接收操作
0	0	不输出奇偶校验位。	没有校验位接收。
0	1	奇偶校验位输出0。	以0校验[注3]的方式接收。
1	0	奇校验输出。	以奇校验的方式进行判断。
1	1	偶校验输出。	以偶校验的方式进行判断。

使能或禁止接收操作

RXE6	
0	禁止接收操作（同步复位接收电路）。
1	接收操作使能。

使能或禁止发送操作

TXE6	
0	禁止发送操作（同步复位发送电路）。
1	发送操作使能。

内部工作时钟操作的使能或禁止

POWER6	
0[注4]	内部工作时钟操作禁止(固定为低电平)且异步复位内部电路[注5]。
1[注6]	内部工作时钟操作使能。

备注　通信执行期间（POWER6 = 1 和 TXE6 = 1，或 POWER6 = 1 和 RXE6 = 1），可通过软件来刷新（写入相同值）ASIM6 寄存器的值。

（‘注’和‘注意事项’在下页给出。）

注　**1.** TXE6 由 CKSR6 寄存器所设置的基准时钟（f_{XCLK6}）同步。为了重新使能发送操作，将 TXE6 设置为 0 并且经过一个基准时钟(f_{XCLK6})之后再将 TXE6 设置为 1。如果在一个基准时钟(f_{XCLK6})逝去之前就将 TXE6 设置为 1，则发送电路不会被初始化。

2. RXE6 由 CKSR6 寄存器所设置的基准时钟（f_{XCLK6}）同步。为了重新使能接收操作，将 RXE6 设置为 0 并且经过一个基准时钟(f_{XCLK6})之后再将 RXE6 设置为 1。如果在一个基准时钟(f_{XCLK6}) 逝去之前就将 RXE6 设置为 1，则接收电路不会被初始化。

3. 如果选择“以 0 校验的方式接收”，则不执行奇偶校验的判断。因此，不设置 ASIS6 寄存器的 PE6 标志且不产生错误中断。

4. 发送操作期间，当 POWER6 设置为 0 时，TxD6 引脚的输出变为高电平并且来自 RxD6 引脚的输入固定为高电平。

5. 重置 ASIS6 寄存器、ASIF6 寄存器、ASICL6 寄存器的 SBRF6 和 SBRT6、以及 RXB6 寄存器。

6. POWER6 设置为 1 并且经过一个基准时钟(f_{XCLK6})之后，基准时钟 (f_{XCLK6})用作内部工作时钟。

注意事项 1. **启动时，通过将 POWER6 设置为 1 后再将 TXE6 设置为 1 来启动发送操作，然后至少等待一个基准时钟(f_{XCLK6})之后，再将发送数据设置至 TXB6 寄存器。若要停止发送操作，先将 TXE6 设置为 0 后，再将 POWER6 设置为 0。**

2. **启动时，先将 POWER6 设置为 1，然后再将 RXE6 设置为 1 且经过一个基准时钟(f_{XCLK6})之后，才能进入接收使能状态。若要停止接收操作，先将 RXE6 设置为 0 后，再将 POWER6 设置为 0。**

3. **在 RxD6 引脚输入高电平的状态下，设置 POWER6 = 1 → RXE6 = 1。如果在低电平输入期间设置 POWER6 = 1 → RXE6 = 1，也能启动接收操作，但是无法接收正确数据。**

4. **重写 PS61、PS60 和 CL6 之前，将 TXE6 和 RXE6 清为 0。**

5. **LIN 通信操作期间，将 PS61 和 PS60 固定为 0。**

6. **将 TXE6 设置为 0 之后再对 SL6 进行重写，因为接收操作始终在“停止位位数 = 1”的情况下执行，所以接收操作与 SL6 的设置值无关。**

7. **将 RXE6 设置为 0 后再对 ISRM6 进行重写。**

(4) ASICL6 寄存器的设置

该寄存器用于控制串行接口 UART6 的串行通信工作。

图 4-4. 异步串行接口控制寄存器 6 (ASICL6)的格式

ASICL6

SBRF6	SBRT6	SBTT6	SBL62	SBL61	SBL60	DIR6	TXDLV6

使能或禁止TxD6的输出反转

TXDLV6	
0	正常TxD6输出。
1	反转TxD6输出。

起始位说明

DIR6	
0	MSB先行传输。
1	LSB先行传输。

SBF发送的输出宽度控制

SBL62	SBL61	SBL60	
1	0	1	输出13位长度的SBF。
1	1	0	输出14位长度的SBF。
1	1	1	输出15位长度的SBF。
0	0	0	输出16位长度的SBF。
0	0	1	输出17位长度的SBF。
0	1	0	输出18位长度的SBF。
0	1	1	输出19位长度的SBF。
1	0	0	输出20位长度的SBF。

SBF发送触发

SBTT6	
0	–
1	SBF发送触发。

SBF接收触发

SBRT6	
0	–
1	SBF接收触发。

SBF接收状态标志(只读)

SBRF6	
0	如果POWER6和RXE6都设置为0，或者如果SBF的接收正确完成。
1	正在进行SBF接收。

备注　如果数据 0 已通过 SBRT6 和 SBTT6 写至 ASICL6 中，通信执行期间（POWER6 = 1 和 TXE6 = 1，或 POWER6 = 1 和 RXE6 = 1），可通过软件来刷新（写入相同值）ASICL6 寄存器的值。

（‘注意事项’在下页给出。）

注意事项 **1.** **LIN** 通信操作时，**SBF** 发送和 **SBF** 接收使用该接口。有关 **SBF** 发送和 **SBF** 接收之详情，参见各产品（**78K0S/KA1+**、**78K0S/KB1+**）的用户手册。

2. 在 **SBF** 接收错误情况下，操作将又返回到 **SBF** 接收模式。**SBRF6** 标志的状态将保持为**(1)**。

3. 在将 **POWER6** 和 **RXE6** 设置为 **1** 之后，将 **SBRT6** 设置为 **1**。而且，**SBRT6** 设置为 **1** 后，在 **SBF** 接收结束（产生中断请求信号）之前不要将 **SBRT6** 清为 **0**。

4. **SBRT6** 的读取值始终为 **0**。**SBF** 接收正确完成后，**SBRT6** 自动清为 **0**。

5. 在将 **POWER6** 和 **TXE6** 设置为 **1** 之后，将 **SBTT6** 设置为 **1**。而且，**SBTT6** 设置为 **1** 后，**SBF** 发送结束（产生中断请求信号）之前不要将 **SBTT6** 清为 **0**。

6. **SBTT6** 的读取值始终为 **0**。**SBF** 发送完成后，**SBTT6** 自动清为 **0**。

7. 在将 **TXE6** 和 **RXE6** 清为 **0** 之后，重写 **DIR6** 和 **TXDLV6**。

(5) TXB6 寄存器的操作

本寄存器用来设置串行接口 UART6 的发送数据。通过将发送数据写至 TXB6 寄存器来启动发送操作。

如果数据长度设为 7 位：

- 当 LSB 先行传输时，传输 TXB6 的位 0 至位 6，且不发送 TXB6 的 MSB 位。
- 当 MSB 先行传输时，传输 TXB6 的位 7 至位 1，且不发送 TXB6 的 LSB 位。

图 **4-5.** 发送缓冲器寄存器 **6 (TXB6)**的格式

TXB6

MSB LSB

注意事项 **1.** 为了启动发送操作，在将 **TXE6** 设置为 **1** 并且至少等待一个基准时钟(**f_{XCLK6}**)之后，将发送数据写至 **TXB6** 中。

2. 当 **ASIF6** 寄存器的 **TXBF6** 为 **1** 时，不要向 **TXB6** 寄存器中写入数据。

3. 通信执行期间（**POWER6 = 1** 和 **TXE6 = 1**，或 **POWER6 = 1** 和 **RXE6 = 1**），不要通过软件刷新（写入相同值）**TXB6** 寄存器的值。顺序发送期间为了输出相同的值，**TXB6** 寄存器写入相同值之前，确保其值为 **0**。

[专栏] 什么是 LIN?

LIN 表示 Local Interconnect Network，是一种低速（1 至 20 kbps）串行通信协议，旨在于降低汽车网络的成本。串行接口 UART6 支持 LIN 总线。

LIN 通信为单主机通信，一台主机最多可连接 15 台从机。

LIN 从机用于控制开关、传动装置和传感器，且这些装置可通过 LIN 网络连接到 LIN 主机上。

通常，LIN 主机接入诸如 CAN（Controller Area Network）之类的网络。

另外，LIN 总线使用单线方式且由遵循 ISO9141 规范的收发器连接至各个节点。

在 LIN 协议中，主机发送带有波特率信息的帧。从机接收由主机发送的帧并校正与主机的波特率误差。如果主机和从机的波特率误差为±15%或更少，则可以得到校正。

(6) RXB6 寄存器的操作

本缓冲器寄存器用于存储串行接口 UART6 的接收数据。

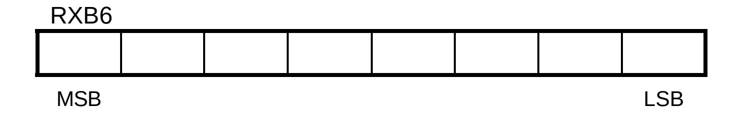
一次接收一个字节的数据，传输新接收的数据。

如果数据长度设为 7 位：

- 当 LSB 在先接收时，将接收数据传输到 RXB6 的位 0 至位 6 且 RXB6 的 MSB 位始终为 0。
- 当 MSB 在先接收时，将接收数据传输到 RXB6 的位 7 至位 1 且 RXB6 的 LSB 位始终为 0。

如果发生溢出错误（OVE6），则接收数据将不会传输到 RXB6 寄存器中。

图 **4-6.** 接收缓冲器寄存器 **6 (RXB6)**的格式



注意事项 **1.** 在将 **RXE6** 设置为 **1** 并且经过一个基准时钟(**f**$_{XCLK6}$)之后，进入接收使能状态。

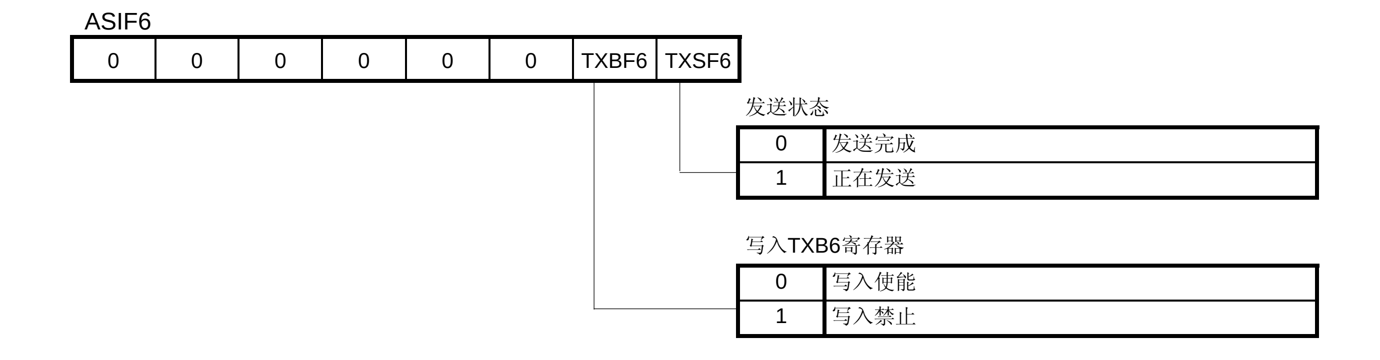
2. 即使出现接收错误，也要读取 **RXB6** 寄存器的值。否则，接收后续数据时会产生溢出错误，并且该接收错误状态将持续存在。

(7) ASIF6 寄存器的操作

本只读寄存器指示了串行接口 UART6 的发送状态。

即使在中断服务期间，仍然能够连续发送而不间断，利用 ASIF6 寄存器确认 TXB6 寄存器的数据传输状态，并且将下一个数据写到 TXB6 寄存器中。

图 **4-7.** 异步串行接口发送状态寄存器 **6 (ASIF6)**的格式



注意事项 **1.** 为了连续发送数据，将第一个发送数据（第一个字节）写至 **TXB6** 寄存器；确保在证实 **TXBF6** 位的值为“**0**”之后，再将下一个发送数据（第二个字节）写至 **TXB6** 寄存器。如果当 **TXBF6** 位的值为“**1**”时，将数据写至 **TXB6** 寄存器，则不能保证数据发送。

2. 连续发送完成后，确保在发送完成的中断产生之后 **TXSF6** 位为“**0**”时，初始化发送单元。如果当 **TXSF6** 位为“**1**”时执行初始化，则不能保证数据发送。

(8) ASIS6 寄存器的操作

该只读寄存器指示串行接口 UART6 接收完成时的错误状态。

图 **4-8.** 异步串行接口接收错误状态寄存器 **6 (ASIS6)**的格式

ASIS6

0	0	0	0	0	PE6	FE6	OVE6

指示溢出错误的状态标志

0	当POWER6 = 0和RXE6 = 0时，或者当读取ASIS6寄存器的值时。
1	当接收数据设置至RXB6寄存器，且在读取该数据之前下一次接收操作已完成（溢出错误）时。

指示帧错误的状态标志

0	当POWER6 = 0和RXE6 = 0时，或者当读取ASIS6寄存器的值时。
1	接收完成时，没有检测到停止位（帧错误）。

指示奇偶校验错误的状态标志

0	当POWER6 = 0和RXE6 = 0时，或者当读取ASIS6寄存器的值时。
1	当发送数据的奇偶校验位与接收完成时的奇偶位不相符（奇偶校验错误）时。

注意事项 **1. PE6** 的作用根据 **ASIM6** 寄存器位 **PS61** 和位 **PS60** 设置值的不同而异。

2. 仅检查接收数据的第一个停止位，而与停止位的位数无关。

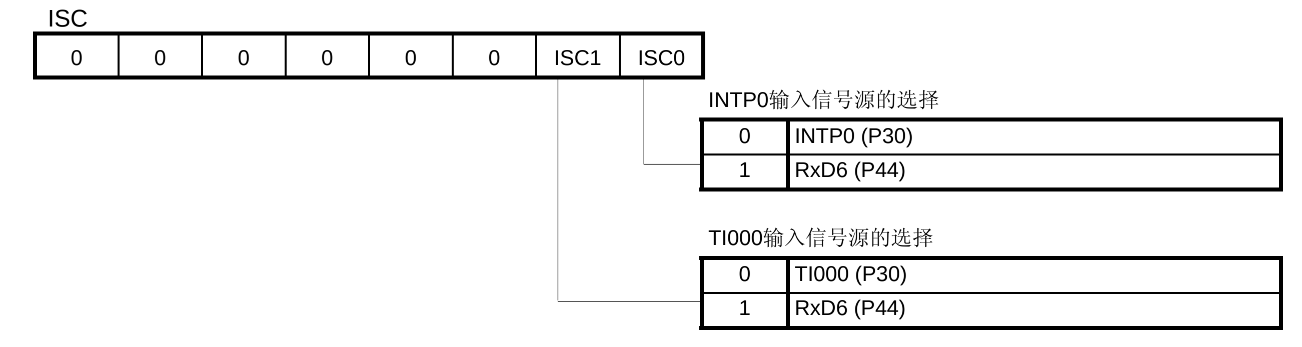
3. 如果产生溢出错误，则下一个接收数据会被丢弃而不是写入 **RXB6** 寄存器。

4. 确保在读取 **RXB6** 寄存器的值之前读取 **ASIS6** 寄存器的值。

(9) ISC 寄存器的设置

在 LIN 接收期间，当接收由主机发送的状态信号时，设置该寄存器。通过设置 ISC 寄存器，RxD6 引脚的输入信号可输入到外部中断 (INTP0)和 16 位定时器/事件计数器 00 中。

图 **4-9.** 输入开关控制寄存器**(ISC)**的格式



注意事项 有关**LIN**发送和**LIN**接收之详情，参见各产品（**78K0S/KA1+**、**78K0S/KB1+**）的用户手册。

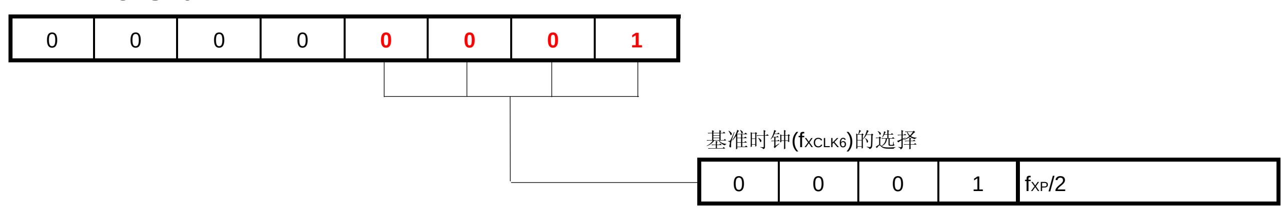
[实例]　如下设置串行接口UART6来启动串行发送和接收操作：

- 波特率：　9，600 bps。
- 数据字符长度：　7位。
- 奇偶校验位规约：　偶校验。
- 停止位位数：　1位。
- 起始位：　LSB先行传输。
- TxD6输出：　正常输出。
- 基于错误的发生而产生中断：　INTSRE6。

（提供至外围硬件的时钟振荡频率(f_{XP}) = 8 MHz，不执行LIN通信。）
（与本举例程序中的源程序内容一致。）

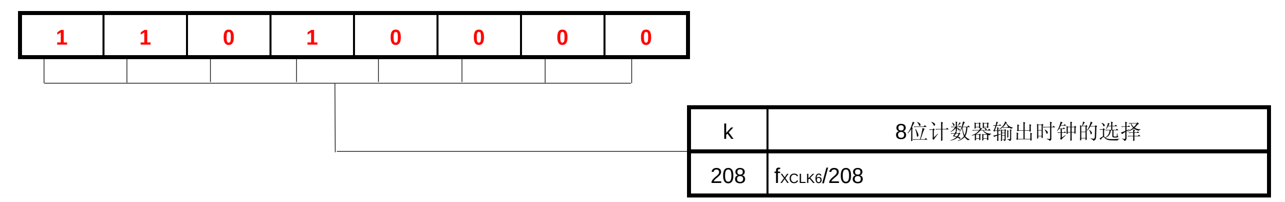
(1) 寄存器设置

<1>　CKSR6



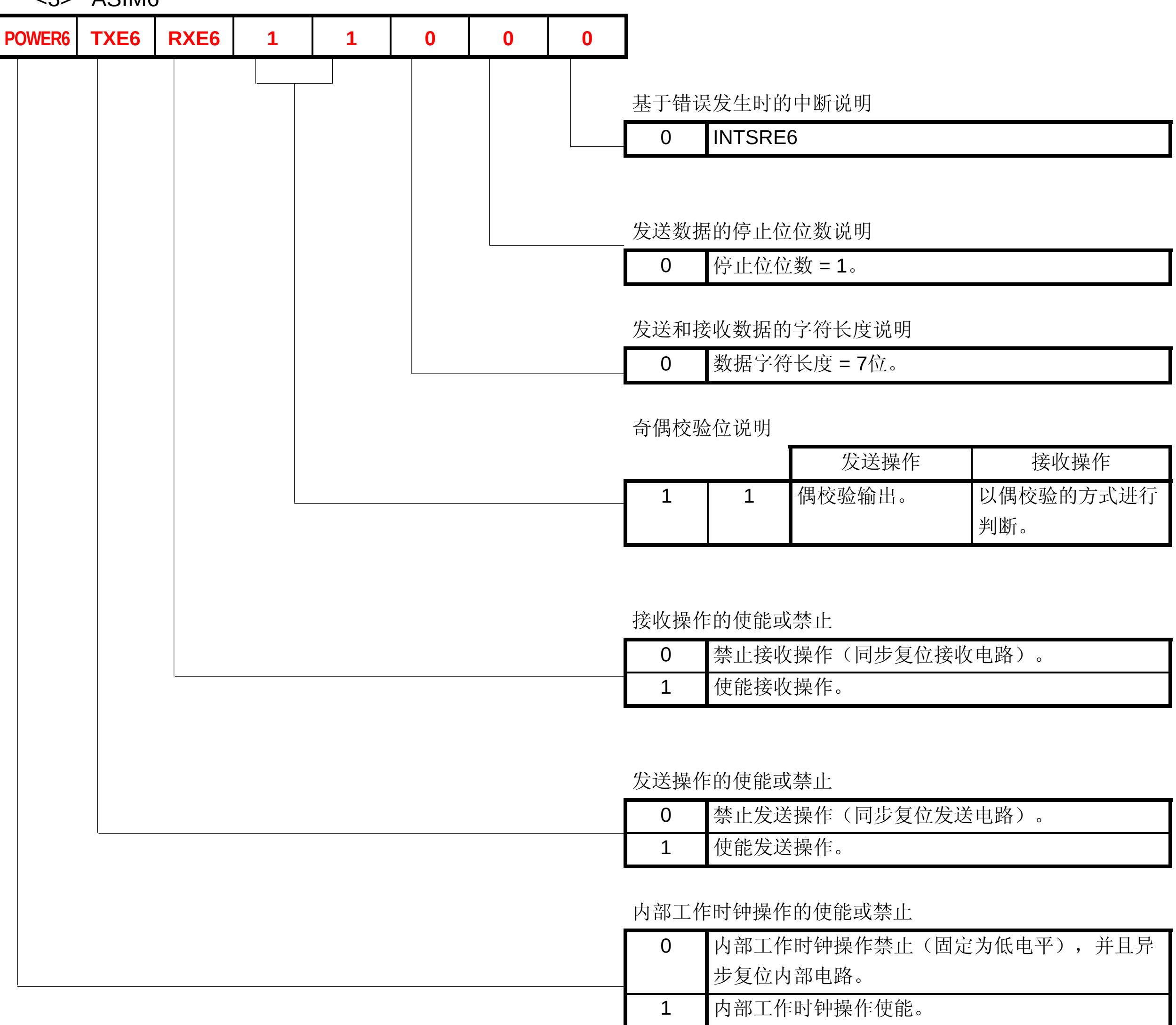
- 基准时钟(f_{XCLK6}) = f_{XP}/2 = 8 MHz/2 = 4 MHz

<2>　BRGC



- 波特率 = $\frac{f_{XCLK6}}{2 \times k}$ [bps] = $\frac{4\ MHz}{2 \times 208}$ [bps] = $\frac{4,000,000}{2 \times 208}$ [bps] $\cong$ 9,600 [bps]

<3> ASIM6



<4> ASICL6：依照缺省设置（起始位：LSB 先行传输、TxD6 输出：正常输出）

<5> PMx、 Px

PM43	P43	PM44	P44	UART6的操作	引脚功能	
					TxD6/INTP1/P43	RxD6/P44
0	1	1	×	发送和接收	TxD6	RxD6

备注　×：不必理会。

(2) 举例程序

<1>　汇编语言

```
SET1    P4.3
CLR1    PM4.3
SET1    PM4.4
MOV     CKSR6，        #1
MOV     BRGC6，        #208
MOV     ASIM6，        #00011000B
SET1    POWER6
SET1    TXE6
SET1    RXE6
```

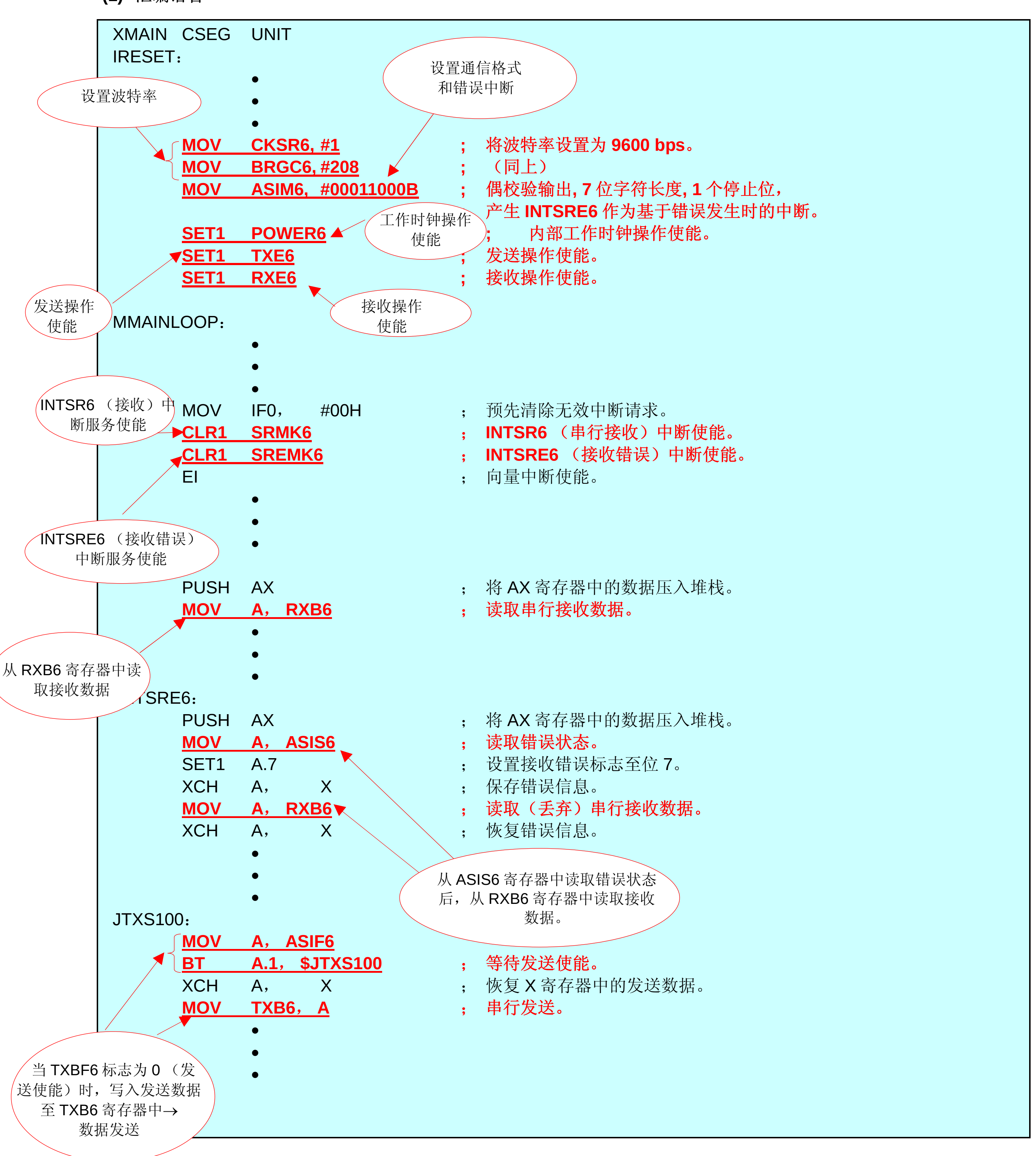
<2>　C 语言

```
P4.3 = 1；
PM4.3 = 0；
PM4.4 = 1；
CKSR6 = 1；
BRGC6 = 208；
ASIM6 = 0b00011000；
POWER6 = 1；
TXE6 = 1；
RXE6 = 1；
```

[摘自本举例程序中的源程序部分]

从附录 A 程序清单中摘录与串行接口 UART6 功能相关的程序，如下所示（同上述 [实例] 中的内容一致）：

(1) 汇编语言



(2)　C 语言

```
void hdwinit(void){
        unsigned char ucCnt200us;    /*用于 200 us 等待的 8 位变量*/
                    •
                    •
                    •
    CKSR6 = 1;                       /*   将波特率设置为 9600 bps */
    BRGC6 = 208;                     /*   （同上） */
    ASIM6 = 0b00011000;              /*   偶校验输出，7 位字符长度， 1 个停止位 */
                                          产生 INTSRE6 中断作为基于错误发生的中断 */
    POWER6 = 1;                      /*   内部工作时钟操作使能 */
    TXE6  =   1;                     /*   发送操作使能 */
    RXE6  =   1;                     /*   接收操作使能 */
    return;
}

void main(void)
{
                    •
                    •
                    •
    IF0 = 0x00;                      /*   预先清除无效中断请求 */
    SRMK6 = 0;                       /*   INTSR6（串行接收）中断使能 */
    SREMK6 = 0;                      /*   INTSRE6（接收错误）中断使能 */
    EI();                            /*   中断向量使能 */
                    •
                    •
                    •
}

__interrupt void fn_intsr6()
{
    unsigned char ucData;
    ucData = RXB6;                   /*   读取串行接收数据 */
                    •
                    •
                    •
}

__interrupt void fn_intsre6()
{
    unsigned char ucData;
    unsigned char ucTemp;
    ucData = ASIS6 | 0b10000000;       /*   通过设置错误标志至位 7 存储错误信息 */
    ucTemp = RXB6;                     /*   读取（丢弃）串行接收数据 */
                    •
                    •
                    •
}

void fn_uart_send(unsigned char ucTxData)
{
    g_ucAsif6 = ASIF6;                   /*   读取发送状态 */
        while (g_ucAsif6.1)              /*   等待发送使能 */
    {
        g_ucAsif6 = ASIF6;               /*   读取发送状态 */
    }
    TXB6 = ucTxData;                     /*   串行发送 */
    return;
}
```

4.2 接收数据或接收的错误内容并发送数据

本举例程序中，利用串行接口 UART6 执行串行通信，并且根据接收的数据或接收的错误内容发送数据。假定接收数据为 1 个字符的 ASCII 码且发送数据为 4 个字符的 ASCII 码。

当接收操作正常完成时，因为数据字符长度设置为 7 位且起始位为 LSB 位，所以接收数据位顺序传输至接收缓冲器寄存器（RXB6）的起始位（位 0，即 LSB 位）直到位 6，此时，位 7（MSB）始终为 0。中断（INTSR6）服务执行期间，RXB6 寄存器的接收数据将存储到缓冲器中。

当出现接收错误时，中断（INTSRE6）服务执行期间，ASIS6 寄存器的错误状态存储至缓冲器的位 0 至位 6 且位 7 被设为 1。

因此，读取缓冲器时，通过位 7 来确认位 0 至位 6 的数据是接收数据还是错误状态，从而识别位 0 至位 6 数据的内容。

所读取的接收数据或接收错误内容与相应的发送数据对应关系如下：

● 正常接收（位 7 为 0）

1个字符的接收数据（十六进制数）	4个字符的发送数据（十六进制数）			
T（54H）	O（4FH）	K（4BH）	“CR”（0DH）	“LF”（0AH）
t（74H）	o（6FH）	k（6BH）	“CR”（0DH）	“LF”（0AH）
非上述字符	U（55H）	C（43H）	“CR”（0DH）	“LF”（0AH）

● 错误接收（位 7 为 1）

接收错误内容（ASIS6寄存器标志值）	4个字符的发送数据（十六进制数）			
奇偶校验错误（PE6为1）	P（50H）	E（45H）	“CR”（0DH）	“LF”（0AH）
帧错误（FE6为1）	F（46H）	E（45H）	“CR”（0DH）	“LF”（0AH）
溢出错误（OVE6为1）	O（4FH）	E（45H）	“CR”（0DH）	“LF”（0AH）

备注 “CR”和“LF”为控制字符。“CR”和“LF”的组合形成一个换行码。

[专栏] 宏和子程序

本汇编语言举例程序中，宏和子程序用于简化程序代码。因为在处理程序过程中，仅需修订一处即可完成所有相关的处理，并且程序的可读性强，所以利用宏和子程序能够很方便地维护整个程序。

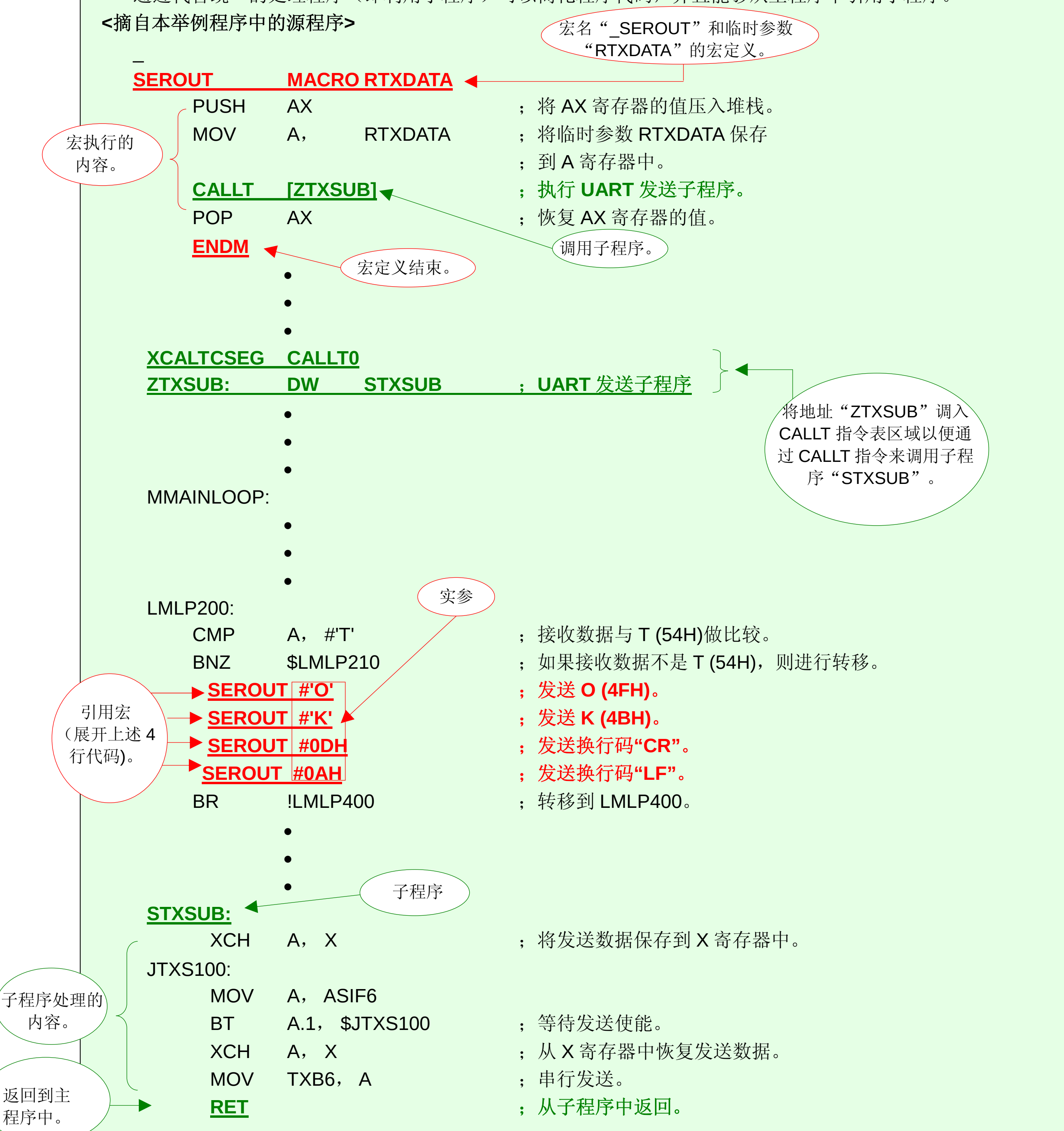
<1> 宏

运用宏来定义频繁用到的处理程序名称，并利用所定义的宏名（标识符）很容易简化程序代码。因为宏引用展开为由宏定义的程序代码，而不需要诸如 CALL、CALLT、及 RET 之类的冗余指令，所以能够缩短程序执行时间。而且，由于参数可被定义，所以通过简单地改变这些参数，即可定义相似的处理程序。

<2> 子程序

通过代替统一的处理程序（即利用子程序）可以简化程序代码，并且能够从主程序中引用子程序。

<摘自本举例程序中的源程序>



第五章　使用器件进行操作检验

利用下载的举例程序，本章介绍了从连编到使用该器件进行操作检验的流程。

5.1　连编举例程序

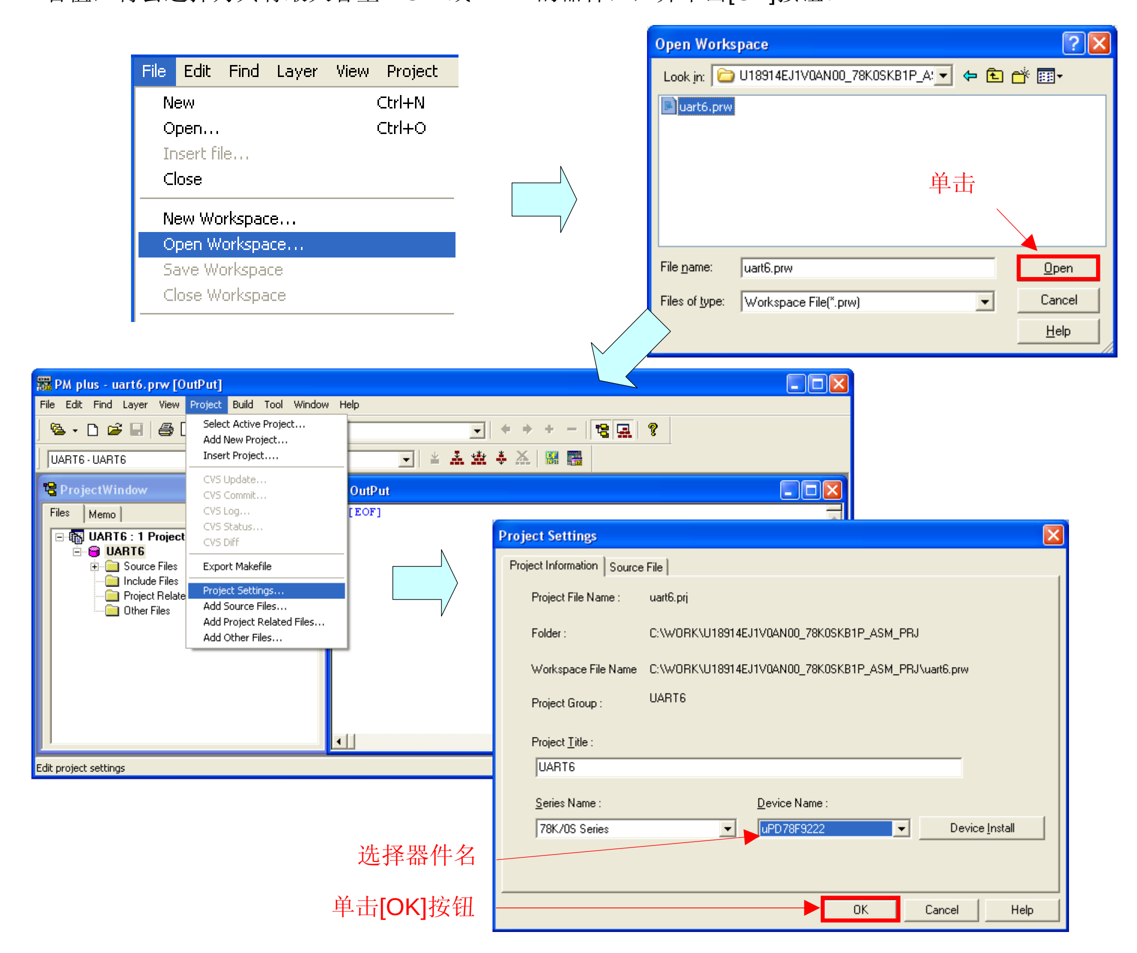
本节介绍了如何对利用单 图标所下载的举例程序（源文件 + 工程文件）进行连编。关于如何编译其它的下载程序，参见 **78K0S/Kx1+举例程序启动指南使用说明**中的**第三章 注册集成开发环境 PM+工程项目和执行编译**。

关于如何操作 PM+之详情，参见 **PM+工程管理器用户手册**。

(1) 启动 PM+。

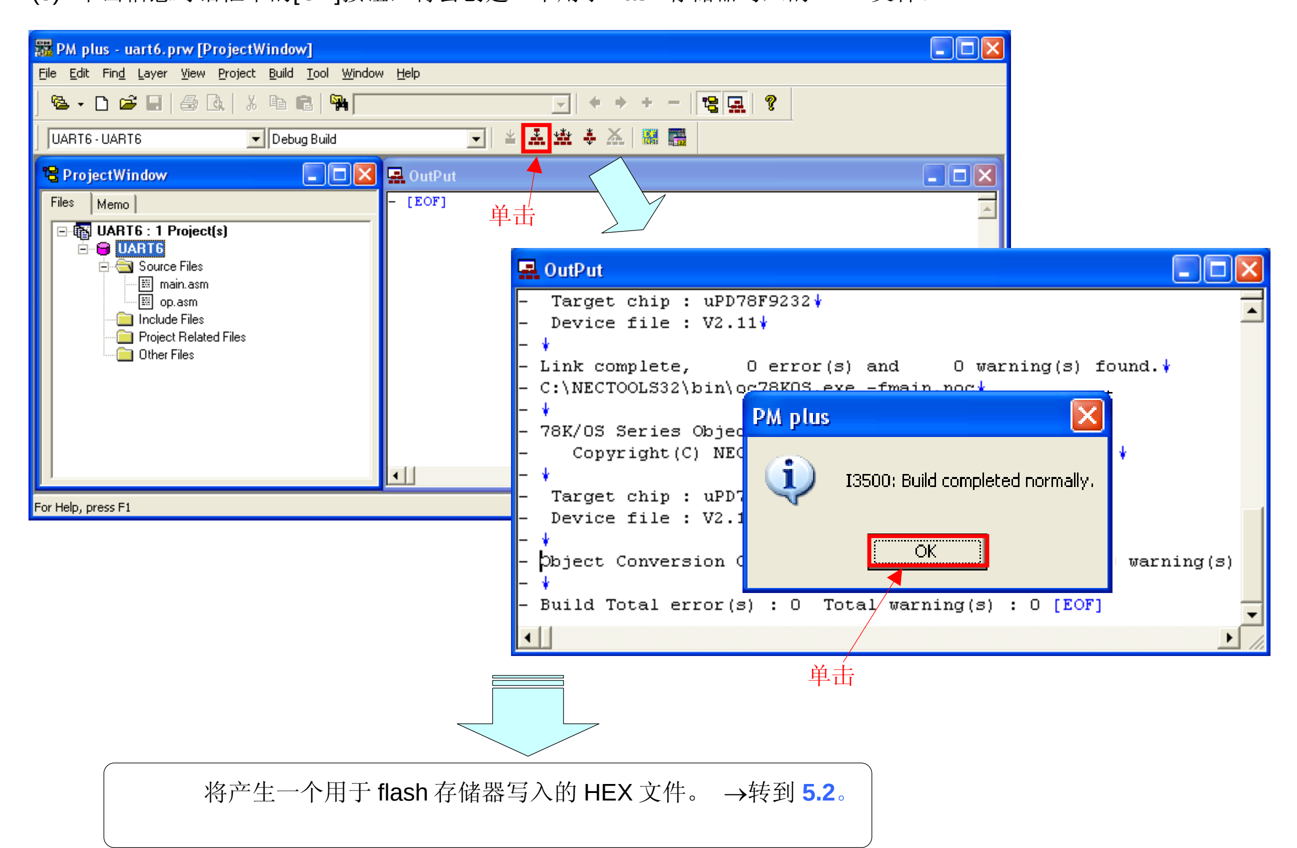
(2) 从[File]菜单中单击[Open Workspace]选项选择“uart6.prw”并单击[Open]按钮。将创建一个工作空间，源文件自动读入。

(3) 从[Project]菜单中选择[Project Settings]选项。当[Project Settings]对话框打开时，选择所用器件的名称（按照缺省值，将会选择为具有最大容量 ROM 或 RAM 的器件），并单击[OK]按钮。



(4) 单击 （[Build]按钮）。当“main.asm”和“op.asm”源文件正常编译后，将显示“I3500: Build completed normally.”信息。

(5) 单击信息对话框中的[OK]按钮，将会创建一个用于 flash 存储器写入的 HEX 文件。





[专栏]编译链接错误

使用 PM+进行编译，当显示“A006 File not found ‘C:\NECTOOLS32\LIB78K0S\s0sl.rel’”或“*** ERROR F206 Segment ‘@@DATA’ can’t allocate to memory - ignored.”错误信息时，按照下列步骤改变编译选项设置。

<1> 从[Tool]菜单中选择[Compiler Options]选项。
<2> 将会显示[Compiler Options]对话框，选择[Startup Routine]选项。
<3> 取消选择[Using Fixed Area of Standard Library]复选框。(其它复选框不变。)

当取消 [Using Fixed Area of Standard Library]复选框时，只允许使用一个位于 RAM 区域内被保护的 118 字节区域作为固定标准库域，但是，禁止使用标准库(例如 getchar 函数和 malloc 函数)。
在本举例程序中，当使用由单击 图标下载的文件时，[Using Fixed Area of Standard Library]复选框缺省为不选择。

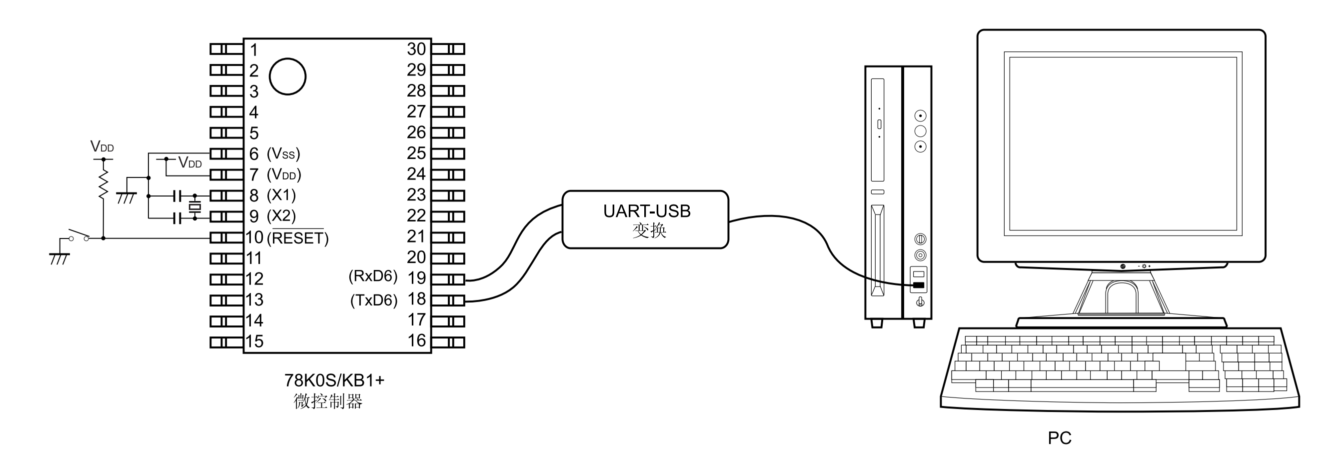
5.2　器件的操作

本节介绍了利用该器件进行操作检验的一个实例。

执行编译所生成的 HEX 文件可以写入该器件的 flash 存储器中。

至于如何写入该器件的 flash 存储器，参见 **78K0S/Kx1+简化的 Flash 写入手册信息**（正在准备中）。

如何连接该器件和所用外围硬件的实例显示如下：



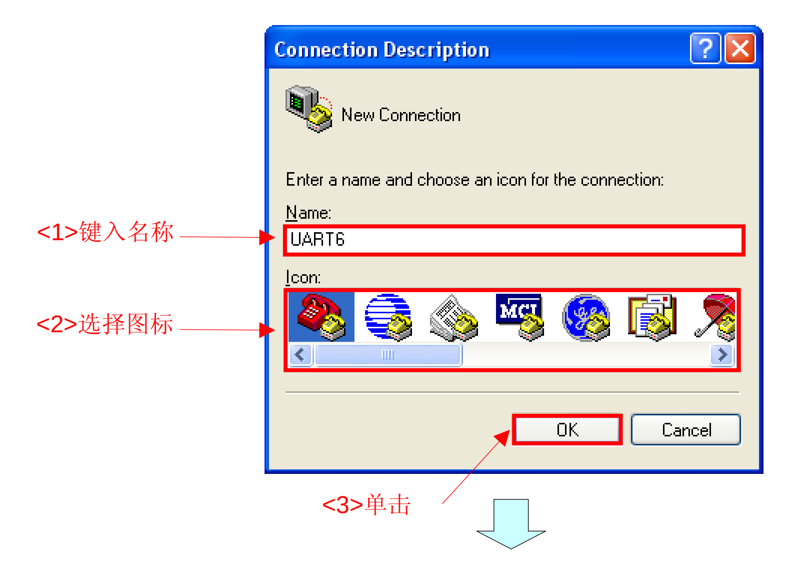
上图所示为一个操作实例，其中，本举例程序已写入所连接的器件，所使用的超级终端是 Windows™ 2000 和 Windows XP™ 提供的标准工具，如下所示：

(1) 如上图所示，连接已写入本举例程序的器件，并按照下列步骤启动超级终端：

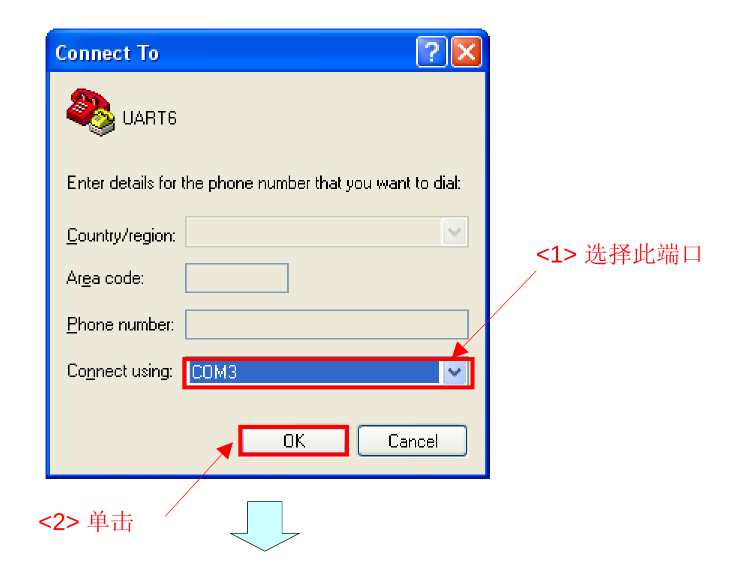
- Windows 2000：按照[Start]、[Programs]、[Accessories]、[Communications]、和[超级终端]的顺序进行选择。

- Windows XP：按照[Start]、[All programs]、[Accessories]、[Communications]、和[超级终端]的顺序进行选择。

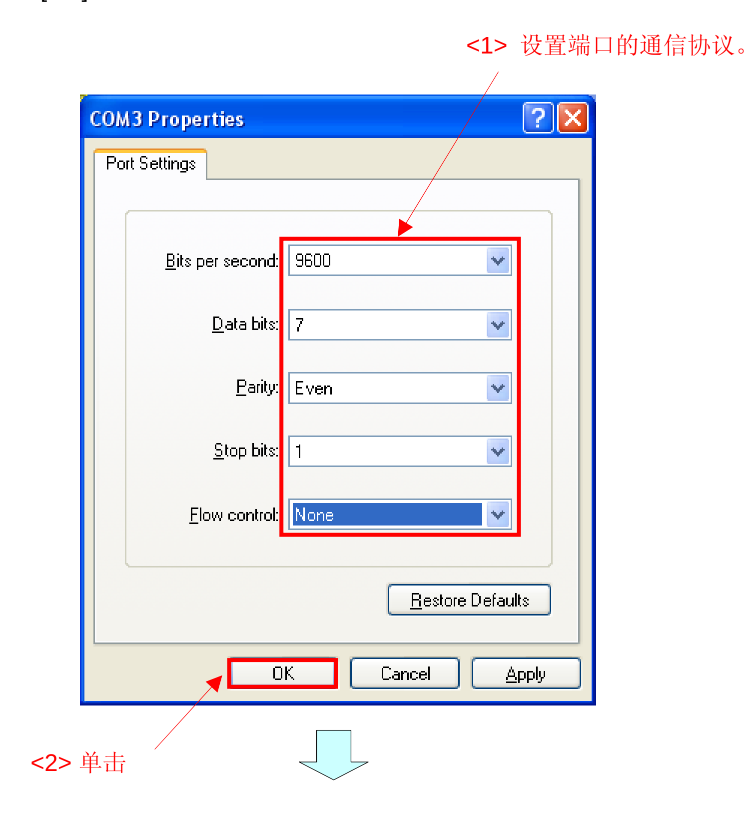
(2) 打开 [Connection Description]对话框。键入任意名称（本实例中键入“UART6”），选择一个图标（本实例中选择最左边的图标），并单击[OK]按钮。



(3) 打开一个新的对话框，选择连接 USB 电缆的端口（本实例中选择“COM3”端口）并单击[OK]按钮。

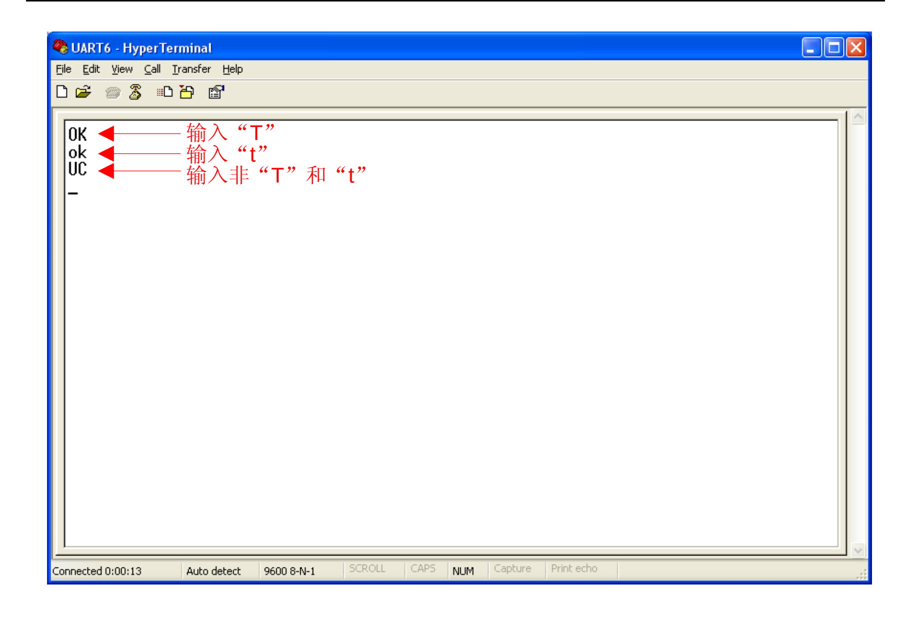


(4) 打开[xxxx Properties]对话框（xxxx：第(3)步中设置的端口名称，本实例中显示[COM3 Properties]）。如下设置端口的通信协议，并单击[OK]按钮。



(5) 打开[yyyy - 超级终端]窗口（yyyy：步骤(2)中设置的通信名称，本实例中显示[UART6 - 超级终端] ）。 根据键盘键入，输入字符将显示在如下所示窗口：

键盘键入	显示
T	OK + “换行”
t	ok + “换行”
非上述字符	UC + “换行”



第六章　相关文档

文档名称		日语/英语
78K0S/KA1+用户手册		PDF
78K0S/KB1+用户手册		PDF
78K/0S 系列指令用户手册		PDF
RA78K0S 汇编语言程序包用户手册	语言	PDF
	操作	PDF
CC78K0S C编译器用户手册	语言	PDF
	操作	PDF
PM+ 工程管理器用户手册		PDF
SM+ 系统仿真器操作用户手册		PDF
78K0S/KA1+ 简化的Flash写入手册 MINICUBE2 信息		PDF
78K0S/Kx1+ 使用说明	举例程序启动指南	PDF
	举例程序（初始设置）LED灯开关控制	PDF
	举例程序（中断）由开关输入产生的外部中断	PDF
	举例程序（低电压检测）电压低于2.7 V时的复位	PDF

作为程序举例清单，78K0S/KB1+微控制器的源程序显示如下：

● main.asm（汇编语言版）

```
;  ****************************************************************************
;
;         NEC Electronics     78K0S/KB1+
;
;  ****************************************************************************
;         78K0S/KB1+     举例程序
;  ****************************************************************************
;         串行接口 UART6
;  ****************************************************************************
; <<历史记录>>
;         2007.8.--           发布
;  ****************************************************************************
;
; <<概要>>
;
; 本举例程序提供了串行接口 UART6 的一个应用实例。
; 将 8 MHz 的晶体振荡时钟或陶瓷振荡时钟作为系统时钟信号源来执行波特率为 9600 bps 的串行通信。
; 假定发送和接收的数据为 ASCII 形式，并且根据 1 个字符的接收数据来发送 4 个字符的数据。
; 当出现接收错误时，也会发送与该错误数据相对应的 4 个字符数据。
;
;   <主要设置内容>
;
;   - 停止看门狗定时器的运行。
;   - 将低电压检测电压（VLVI）设置为 4.3 V +-0.2 V。
;   - VDD >= VLVI 之后，当 VDD < VLVI 时产生内部复位信号（低电压检测器）。
;   - 将 CPU 时钟频率设置为 8 MHz。
;   - 将提供至外围硬件的时钟频率设置为 8MHz。
;   - 设置串行接口 UART6。
;   - 将 DE 寄存器和 HL 寄存器用于中断服务（类似于全局变量）。
;
;   <串行通信协议>
;
;   - 波特率：            9600 bps。
;   - 数据字符长度：      7 位。
;   - 奇偶校验位规约：    偶校验。
;   - 停止位位数：        1 位。
;   - 起始位说明：        LSB 先行传输。
;
;
;   <接收数据>
;
;   因为假定接收的数据为 ASCII 码形式，所以数据字符长度设置为 7 位且设置 LSB 在先传输。
;   接收数据传输至 RXB6 寄存器的位 0 至位 6，并且位 7（MSB）始终为 0。
;   而且，当出现接收错误时，所接收错误信息的位 7 被设为 1 并且存储到用于存储接收数据的缓冲器中。
```

```
;   因此，当读取缓冲器时，根据位 7 的状态来识别缓冲器中的数据是接收数据还是接收错误状态。
;
;
;   <顺序接收>
;
;   因为通过中断已将接收数据存储在缓冲器中，并且接收数据
;   从缓冲器的起始地址处连续累积存储，所以能够实现顺序接收。
;   而且，缓冲器已设定为一个环形缓冲区，接收数据存储到缓
;   冲器的末尾后，又会从缓冲器的起始地址处开始存储。
;   此时，已被读取的缓冲区将继续存储接收数据，而未被读取的
;   缓冲区（当未被读取的数据长度已达到缓冲器的容量时）将丢弃接收数据而不存储。
;   一旦缓冲区的数据被读取就会立即存储接收数据。
;   缓冲器容量由 CBUFFSIZE 定义，并且默认为 50 个字节。
;
;
;   <命令说明>
;
;   - 正常接收
;   +------------------------------------------------------------------+
;   |     接收数据     |            4 个字节的发送数据              |
;   | （Hex 数据）   |                （Hex 数据）                  |
;   |------------------------------------------------------------------|
;   |       T          |   O    |   K    | “CR”   |  “LF”  |
;   |    （54H）       |（4FH）|（4BH）|（0DH）|（0AH）|
;   |------------------------------------------------------------------|
;   |       t          |   o    |   k    | “CR”   |  “LF”  |
;   |    （74H）       |（6FH）|（6BH）|（0DH）|（0AH）|
;   |------------------------------------------------------------------|
;   |    其它数据      |   U    |   C    | “CR”   |  “LF”  |
;   |                  |（55H）|（43H）|（0DH）|（0AH）|
;   +------------------------------------------------------------------+
;   # “CR” + “LF”为换行码。
;
;   - 错误接收
;   +------------------------------------------------------------------+
;   |   错误接收       |             4 个字符发送数据               |
;   |     信息         |               （Hex 数据）                 |
;   |------------------------------------------------------------------|
;   | 奇偶校验错误   |   P    |   E    | “CR”   |  “LF”  |
;   |                  |（50H）|（45H）|（0DH）|（0AH）|
;   |------------------------------------------------------------------|
;   |   帧错误         |   F    |   E    | “CR”   |  “LF”  |
;   |                  |（46H）|（45H）|（0DH）|（0AH）|
;   |------------------------------------------------------------------|
;   |   溢出错误       |   O    |   E    | “CR”   |  “LF”  |
;   |                  |（4FH）|（45H）|（0DH）|（0AH）|
;   +------------------------------------------------------------------+
;   # “CR” + “LF”为换行码。
;
; <<I/O 端口设置>>
;
;   输入： P44
```

```
;   输出： P00-P03、P20-P23、P30-P33、P40-P43、P45-P47、P120-P123、P130
;   # 所有未用端口设置为输出模式。
; ****************************************************************************

; ==============================================================================
;
;         定义标号
;
; ==============================================================================
CBUFFSIZE     EQU     50                    ;  接收数据的缓冲区容量。
; ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;
;         定义宏
;
;  UART 发送处理程序定义了一个用于简化程序代码的宏。
;
; ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
_SEROUT       MACRO           RTXDATA

        PUSH   AX                           ;  将 AX 寄存器的数据压入堆栈。
        MOV    A，      RTXDATA             ;  将临时参数 RTXDATA 保存到 A 寄存器中。
        CALLT  [ZTXSUB]                     ;  执行 UART 发送子程序。
        POP    AX                           ;  恢复 AX 寄存器中的数据。
        ENDM
; ==============================================================================
;
;         向量表
;
; ==============================================================================
XVCT    CSEG    AT      0000H
        DW     IRESET                       ; （00）        RESET
        DW     IRESET                       ; （02）        --
        DW     IRESET                       ; （04）        --
        DW     IRESET                       ; （06）        INTLVI
        DW     IRESET                       ; （08）        INTP0
        DW     IRESET                       ; （0A）        INTP1
        DW     IRESET                       ; （0C）        INTTMH1
        DW     IRESET                       ; （0E）        INTTM000
        DW     IRESET                       ; （10）        INTTM010
        DW     IRESET                       ; （12）        INTAD
        DW     IRESET                       ; （14）        --
        DW     IRESET                       ; （16）        INTP2
        DW     IRESET                       ; （18）        INTP3
        DW     IRESET                       ; （1A）        INTTM80
        DW     IINTSRE6                     ; （1C）        INTSRE6
        DW     IINTSR6                      ; （1E）        INTSR6
        DW     IRESET                       ; （20）        INTST6
; ==============================================================================
;
;         CALLT 表
;
;   利用 CALLT 指令（为一个字节的调用指令）能够使频繁调用子程序的指令代码缩短。
```

```
；这时，利用 DW 伪指令将所需地址录入到该表中，
；以便利用 CALLT 指令来调用 UART 发送子程序 STXSUB。
；利用该地址（ZTXSUB）并声明 CALLT [ZTXSUB] 即可调用相应的子程序。
；
；==========================================================================
XCALT CSEG  CALLT0
ZTXSUB：              DW      STXSUB        ；UART 发送子程序。
；==========================================================================
；
；        定义 RAM
；
；==========================================================================
DRAM1 DSEG  UNIT
RRXBUFTOP：DS      CBUFFSIZE     ；接收数据缓冲区。
RRXBUFEND：

DRAM2 DSEG  SADDR
RRXCNT：              DS      1             ；接收计数变量。

；==========================================================================
；
；        定义内存堆栈区
；
；==========================================================================
DSTK  DSEG  AT      0FEE0H
RSTACKEND：DS      20H               ；内存堆栈区 = 32 字节。
RSTACKTOP：                          ；内存堆栈区的起始地址 = FF00H。

；****************************************************************************
；
；        RESET 后的初始化
；
；****************************************************************************
XMAIN CSEG  UNIT
IRESET：
；-----------------------------------------------------------------------------
；        初始化堆栈指针
；-----------------------------------------------------------------------------
        MOVW AX，     #RSTACKTOP
        MOVW SP，     AX                ；设置堆栈指针。
；-----------------------------------------------------------------------------
；        初始化看门狗定时器
；-----------------------------------------------------------------------------
        MOV    WDTM，        #01110111B    ；停止看门狗定时器的运行。
；-----------------------------------------------------------------------------
；        检测低电压 + 设置时钟频率
；-----------------------------------------------------------------------------

；----- 设置时钟频率 <1> -----
        MOV    PCC，  #00000000B            ；提供至 CPU 的时钟频率：（fcpu） = fxp（= fx/4 = 2
MHz）。
        MOV    LSRCM，       #00000001B    ；停止内部低速振荡器的振荡。
```

```
; -----  检查复位信号源  -----
        MOV     A，     RESF            ；  读取复位信号源。
        BT      A.0，   $HRST300        ；  LVI 复位期间，省略后续 LVI 相关处理并转到 SET_CLOCK。

; -----  设置低电压检测  -----
        MOV     LVIS，  #00000000B      ；  设置低电压检测电平（VLVI）为 4.3 V +-0.2 V。
        SET1    LVION                   ；  低电压检测器操作使能。

        MOV     A，     #40             ；  指定 200 us 的等待计数值。
; -----  等待 200 us  -----
HRST100：
        DEC     A
        BNZ     $HRST100                ；  0.5[us/clk] x 10[clk] x 40[计数值] = 200[us]。

; -----  等待 VDD >= VLVI 时的处理程序  -----
HRST200：
        NOP
        BT      LVIF，  $HRST200        ；  如果 VDD < VLVI，则进行转移。

        SET1    LVIMD                   ；  如此设置以便当 VDD < VLVI 时，产生内部复位信号。

; -----  设置时钟频率 <2>  -----
HRST300：
        MOV     PPCC，          #00000000B      ；  提供至外围硬件的时钟频率（fxp） = fx（= 8 MHz）。
                                        ；  -> 提供至 CPU 的时钟频率（fcpu） = fxp = 8 MHz。

; -----------------------------------------------------------------------------
;       初始化端口 0
; -----------------------------------------------------------------------------
        MOV     P0，    #00000000B      ；  设置 P00-P03 的输出锁存为低电平。
        MOV     PM0，   #11110000B      ；  设置 P00-P03 为输出模式。

; -----------------------------------------------------------------------------
;       初始化端口 2
; -----------------------------------------------------------------------------
        MOV     P2，    #00000000B      ；  设置 P20-P23 的输出锁存为低电平。
        MOV     PM2，   #11110000B      ；  设置 P20-P23 为输出模式。

; -----------------------------------------------------------------------------
;       初始化端口 3
; -----------------------------------------------------------------------------
        MOV     P3，    #00000000B      ；  设置 P30-P33 的输出锁存为低电平。
        MOV     PM3，   #11110000B      ；  设置 P30-P33 为输出模式。

; -----------------------------------------------------------------------------
;       初始化端口 4
; -----------------------------------------------------------------------------
        MOV     P4，    #00001000B      ；  设置 P40-P42 和 P44-P47 的输出锁存为低电平，P43 的输出锁
存为高电平（串行发送时的设置）。
        MOV     PM4，   #00010000B      ；  设置 P40-P43 和 P45-P47 为输出模式， P44 为输入模式。

; -----------------------------------------------------------------------------
;       初始化端口 12
```

```
; ---------------------------------------------------------------------------
        MOV     P12，   #00000000B     ；设置 P120-P123 的输出锁存为低电平。
        MOV     PM12，          #11110000B     ；设置 P120-P123 为输出模式。

; ---------------------------------------------------------------------------
;       初始化端口 13
; ---------------------------------------------------------------------------
        MOV     P13，   #00000001B     ；设置 P130 的输出锁存为高电平。

; ---------------------------------------------------------------------------
;       设置 UART6
; ---------------------------------------------------------------------------
        MOV     CKSR6，         #1          ；将波特率设置为 9600 bps。
        MOV     BRGC6，         #208        ；（同上）。
        MOV     ASIM6，         #00011000B     ；偶校验输出， 7 位字符长度， 1 个停止位，
                                            ；INTSRE6 作为基于错误发生时产生的中断。
        SET1    POWER6                      ；内部工作时钟操作使能。
        SET1    TXE6                        ；发送操作使能。
        SET1    RXE6                        ；接收操作使能。

; ********************************************************************************
;
;       主循环
;
; ********************************************************************************
MMAINLOOP：

; -----  初始化 RAM 和通用寄存器  -----
        MOV     RRXCNT，        #0          ；接收计数值 = 0
        MOVW HL，       #RRXBUFTOP ；初始化缓冲区的起始写入地址。
        MOVW DE，       #RRXBUFTOP ；初始化缓冲区的起始读取地址。

; -----  设置中断  -----
        MOV     IF0，   #00H                ；预先清除无效中断请求。
        CLR1    SRMK6                       ；INTSR6（串行接收）中断使能。
        CLR1    SREMK6                      ；INTSRE6（接收错误）中断使能。
        EI                                  ；中断向量使能。

; -----  等待接收中断  -----
LMLP100：
        CMP     RRXCNT，        #0
        BZ      $LMLP100                    ；如果接收计数值为 0，则进行转移。

        MOV     A，  [DE]                   ；读取数据。
        DEC     RRXCNT                      ；接收计数值减 1。
        BT      A.7，   $LMLP300            ；出现接收错误时，如果位 7 为 1，则转移至处理程序。

; -----  正常接收期间的处理程序  -----
LMLP200：
        CMP     A，  #'T'                   ；接收数据与 T（54H）做比较。
        BNZ     $LMLP210                    ；如果接收数据不是 T（54H），则进行转移。
        _SEROUT         #'O'                ；发送 O（4FH）。
        _SEROUT         #'K'                ；发送 K（4BH）。
```

```
        _SEROUT     #0DH            ； 发送换行码“CR”。
        _SEROUT     #0AH            ； 发送换行码“LF”。
        BR      !LMLP400            ； 转移到 LMLP400。

LMLP210：
        CMP     A，  #'t'           ； 接收数据与 t（74H）做比较。
        BNZ     $LMLP220            ； 如果接收数据不是 t（74H），则进行转移。
        _SEROUT     #'o'            ； 发送 o（6FH）。
        _SEROUT     #'k'            ； 发送 k（6BH）。
        _SEROUT     #0DH            ； 发送换行码“CR”。
        _SEROUT     #0AH            ； 发送换行码“LF”。
        BR      !LMLP400            ； 转移到 LMLP400。

LMLP220：
        _SEROUT     #'U'            ； 发送 U（55H）。
        _SEROUT     #'C'            ； 发送 C（43H）。
        _SEROUT     #0DH            ； 发送换行码“CR”。
        _SEROUT     #0AH            ； 发送换行码“LF”。
        BR      !LMLP400            ； 转移到 LMLP400。

；----- 出现接收错误时的处理程序 -----
LMLP300：
        BF      A.2，  $LMLP310     ； 如果没有奇偶校验错误，则进行转移。
        _SEROUT     #'P'            ； 发送 P（50H）。
        _SEROUT     #'E'            ； 发送 E（45H）。
        _SEROUT     #0DH            ； 发送换行码“CR”。
        _SEROUT     #0AH            ； 发送换行码“LF”。

LMLP310：
        BF      A.1，  $LMLP320     ； 如果没有帧错误，则进行转移。
        _SEROUT     #'F'            ； 发送 F（46H）。
        _SEROUT     #'E'            ； 发送 E（45H）。
        _SEROUT     #0DH            ； 发送换行码“CR”。
        _SEROUT     #0AH            ； 发送换行码“LF”。

LMLP320：
        BF      A.0，  $LMLP400     ； 如果没有溢出错误，则进行转移。
        _SEROUT     #'O'            ； 发送 O（4FH）。
        _SEROUT     #'E'            ； 发送 E（45H）。
        _SEROUT     #0DH            ； 发送换行码“CR”。
        _SEROUT     #0AH            ； 发送换行码“LF”。

；----- 更新读取地址 -----
LMLP400：
        INCW    DE                  ； 读取地址加 1。
        MOVW    AX，    DE
        CMPW    AX，    #RRXBUFEND
        BC      $LMLP450            ； 如果读取地址在缓冲区内，则进行转移。
        MOVW    DE，    #RRXBUFTOP  ； 初始化缓冲区的起始读取地址。
LMLP450：
        BR      !LMLP100            ； 转移到 LMLP100。

；******************************************************************************
```

```
;
;        串行接收中断 INTSR6
;
; ****************************************************************************
IINTSR6：
        PUSH   AX                        ；将 AX 寄存器的数据压入堆栈。

; -----  读取接收数据  -----
        MOV    A， RXB6                   ； 读取串行接收数据。

; -----  检查缓冲区空闲存储空间  -----
        CMP    RRXCNT，      #CBUFFSIZE   ； 接收数据长度与缓冲区容量做比较。
        BNC    $HSR100                   ； 如果缓冲区中没有空闲存储空间可用，则不存储数据。
        INC    RRXCNT                    ； 接收计数值加 1。

; -----  存储数据并更新写入地址  -----
        MOV    [HL]，   A                ； 存储接收数据。
        INCW   HL                        ； 写入地址加 1。
        MOVW   AX，     HL
        CMPW   AX，     #RRXBUFEND
        BC     $HSR100                   ； 如果写入地址在缓冲区内，则进行转移。
        MOVW   HL，     #RRXBUFTOP        ； 初始化缓冲区的起始写入地址。

HSR100：
        POP    AX                        ； 恢复 AX 寄存器的数据。
        RETI                             ； 从中断服务中返回。

; ****************************************************************************
;
;        接收错误中断 INTSRE6
;
; ****************************************************************************
IINTSRE6：
        PUSH   AX                        ； 将 AX 寄存器的值压入堆栈。

; -----  读取错误状态  -----
        MOV    A， ASIS6                  ； 读取错误状态。
        SET1   A.7                       ； 将接收错误标志设置至位 7。
        XCH    A，      X                ； 存储错误信息。
        MOV    A， RXB6                   ； 读取（丢弃）串行接收数据。
        XCH    A，      X                ； 恢复错误信息。

; -----  检测缓冲区空闲存储空间  -----
        CMP    RRXCNT，      #CBUFFSIZE   ； 接收计数值与缓冲区容量做比较。
        BNC    $HSRE100                  ； 如果缓冲器中没有空闲存储空间可用，则不保存数据。
        INC    RRXCNT                    ； 接收计数值加 1。

; -----  存储数据并更新写入地址  -----
        MOV    [HL]，   A                ； 存储错误状态。
        INCW   HL                        ； 写入地址加 1。
        MOVW   AX，     HL
        CMPW   AX，     #RRXBUFEND
        BC     $HSR100                   ； 如果写入地址在缓冲器内，则进行转移。
```

```
        MOVW HL，    #RRXBUFTOP ；  初始化缓冲区的起始写入地址。

HSRE100：
        POP     AX              ；  恢复 AX 寄存器的数据。
        RETI

；=============================================================================
；       串行数据发送的子程序
；
；该子程序用于发送串行数据。
；根据如下设置的子程序可发送#DATA 所表示的 1 个字节数据。
；
；       MOV     A，     #DATA ；将 DATA 存储到 A 寄存器中。
；       CALL    !STXSUB        ；  发送 DATA
；=============================================================================
STXSUB：
        XCH     A，     X       ；  将发送数据保存到 X 寄存器中。

；----- 发送使能的等待 -----
JTXS100：
        MOV     A， ASIF6
        BT      A.1， $JTXS100        ；等待发送使能。

；----- 发送数据 -----
        XCH     A，     X       ；  从 X 寄存器中恢复发送数据。
        MOV     TXB6， A        ；  串行发送。

        RET                     ；  从子程序中返回。

end
```

● main.c（C 语言版）

```
/******************************************************************************

        NEC Electronics     78K0S/KB1+

*******************************************************************************
        78K0S/KB1+     举例程序
*******************************************************************************
        串行接口 UART6
*******************************************************************************
<<历史记录>>
        2007.8.--          发布
*******************************************************************************

<<概要>>

本举例程序提供了一个串行接口 UART6 的应用实例。
利用 8 MHz 的晶体振荡时钟或陶瓷振荡时钟作为系统时钟信号源
来执行波特率为 9600 bps 的串行通信。
假定发送和接收的数据为 ASCII 码形式，并且根据所接收的 1 个字符数据来发送 4 个字符数据。
当出现接收错误时，也会根据该错误数据来发送相应的 4 个字符的数据。


 <主要设置内容>

 - 声明由中断运行的函数： INTSR6 -> fn_intsr6（）。
 - 声明由中断运行的函数： INTSRE6 -> fn_intsre6（）。
 - 停止看门狗定时器的运行。
 - 设置低电压检测电压（VLVI）为 4.3 V +-0.2 V。
 - VDD >= VLVI 之后，当 VDD < VLVI 时，产生内部复位信号（低电压检测器）。
 - 将 CPU 时钟频率设置为 8 MHz。
 - 将提供至外围硬件的时钟频率设置为 8 MHz。
 - 设置串行接口 UART6。


 <设置通信协议>

 - 波特率：               9600 bps。
 - 数据字符长度：         7 位。
 - 奇偶校验说明：         偶校验。
 - 停止位位数：           1 位。
 - 起始位说明：           LSB 在先传输。


 <接收数据>

 因为假定的接收数据为 ASCII 码形式，所以数据字符长度设置为 7 位且设置 LSB 先行传输。
 因此，接收数据传输到 RXB6 寄存器的位 0 至位 6，并且位 7 （MSB）始终为 0。
 此外，当出现接收错误时，表示接收错误信息的位 7 被设为 1，
 并且也存储到接收数据所存储的缓冲器中。
 于是，当读取缓冲器中的数据时，根据位 7 来识别缓冲器中的数据是接收数据还是接收错误信息。
```

```
<顺序接收>

因为利用中断接收的数据已存储在缓冲器中，且从缓冲器的起始地址处连续存储，
所以能够进行顺序接收。
而且，缓冲器已设定为一个环形缓冲区，并且当存储到缓冲器的末尾地址后，接收数据又会从缓冲器的
起始地址处开始存储。
此时，接收数据将存储在已被读取的缓冲区中，
但是，未被读取的缓冲区（当未被读取的数据长度已达到缓冲
器的最大容量）将丢弃接收数据而无法存储。
一旦缓冲区的数据被读取，就会立即存储接收数据。
缓冲区的最大容量由 BUFF_SIZE 定义，且默认值为 50 个字节。

<命令说明>

- 正常接收
+-----------------------------------------------------------------------+
|   接收数据  |              4 个字符的发送数据                          |
|（Hex 数据）|                （Hex 数据）                               |
|-----------------------------------------------------------------------|
|       T     |    O    |    K    |   “CR”   |   “LF”   |
|   （54H）   | （4FH） | （4BH） | （0DH）  | （0AH）  |
|-----------------------------------------------------------------------|
|       t     |    o    |    k    |   “CR”   |   “LF”   |
|   （74H）   | （6FH） | （6BH） | （0DH）  | （0AH）  |
|-----------------------------------------------------------------------|
|  其它数据   |    U    |    C    |   “CR”   |   “LF”   |
|             | （55H） | （43H） | （0DH）  | （0AH）  |
+-----------------------------------------------------------------------+
# “CR” + “LF”为换行码。

- 错误接收
+-----------------------------------------------------------------------+
|  错误接收   |          4 个字符的发送数据                              |
|    信息     |            （Hex 数据）                                   |
|-----------------------------------------------------------------------|
|奇偶校验错误|    P    |    E    |   “CR”   |   “LF”   |
|             | （50H） | （45H） | （0DH）  | （0AH）  |
|-----------------------------------------------------------------------|
|   帧错误    |    F    |    E    |   “CR”   |   “LF”   |
|             | （46H） | （45H） | （0DH）  | （0AH）  |
|-----------------------------------------------------------------------|
|  溢出错误   |    O    |    E    |   “CR”   |   “LF”   |
|             | （4FH） | （45H） | （0DH）  | （0AH）  |
+-----------------------------------------------------------------------+
# “CR” + “LF”为换行码。


<<I/O 端口设置>>

 输入： P44
 输出： P00-P03、P20-P23、P30-P33、P40-P43、P45-P47、P120-P123、P130
 # 设置所有未用端口为输出模式。

*******************************************************************************/
```

```
/*==============================================================================

        预处理指示（#pragma）

==============================================================================*/
#pragma     SFR                         /* SFR 名能够在 C 语言层面上进行描述 */

#pragma     EI                          /* EI 指令能够在 C 语言层面上进行描述 */

#pragma     NOP                         /* NOP 指令能够在 C 语言层面上进行描述 */

#pragma interrupt INTSR6 fn_intsr6      /* 中断函数声明：INTSR6 */
#pragma interrupt INTSRE6 fn_intsre6/* 中断函数声明：INTSRE6 */

#define BUFF_SIZE 50                    /* 接收数据的缓冲区容量 */

/*==============================================================================

        函数原型声明

==============================================================================*/

void fn_uart_send（unsigned char ucTxData）；         /* 用于串行数据发送的函数 */

/*==============================================================================

        定义全局变量

==============================================================================*/
static unsigned char g_ucRxBuff[BUFF_SIZE]；  /* 接收数据缓冲器表 */
sreg unsigned char g_ucRxCnt；                /* 用于接收计数值的 8 位变量 */
sreg unsigned char g_ucStoreAddr；      /* 用于写入地址的 8 位变量 */
sreg unsigned char g_ucReadAddr；       /* 用于读取地址的 8 位变量 */
sreg unsigned char g_ucRxData；         /* 用于识别接收数据的 8 位变量 */
sreg unsigned char g_ucAsif6；          /* 用于识别发送状态的 8 位变量 */

/******************************************************************************

        RESET 后的初始化

******************************************************************************/

void hdwinit（void）{
        unsigned char ucCnt200us；      /* 用于等待 200 us 的 8 位变量 */

/*----------------------------------------------------------------------------
        初始化看门狗定时器 + 检测低电压 + 设置时钟频率
----------------------------------------------------------------------------*/

        /* 初始化看门狗定时器 */
        WDTM  = 0b01110111；              /* 停止看门狗定时器的运行 */

        /* 设置时钟频率 <1> */
        PCC   = 0b00000000；        /* 提供至 CPU 的时钟频率（fcpu） = fxp（= fx/4 = 2 MHz） */
```

```
	LSRCM = 0b00000001；		/* 停止内部低速振荡器的振荡 */

	/* 检查复位信号源 */
	if（!（RESF & 0b00000001））{		/* LVI 复位期间，省略后续 LVI 相关处理程序 */

		/* 设置低电压检测 */
		LVIS  = 0b00000000；	/* 设置低电压检测电平（VLVI）为 4.3 V +-0.2 V */

		LVION = 1；		/* 低电压检测器操作使能 */

		for（ucCnt200us = 0； ucCnt200us < 9； ucCnt200us++）{		/* 等待大约 200 us */
			NOP（）；
		}

		while（LVIF）{ /* 等待 VDD >= VLVI */
			NOP（）；
		}

		LVIMD = 1；		/* 如此设置以便当 VDD < VLVI 时，产生内部复位信号 */

	}

	/* 设置时钟频率 <2> */
	PPCC  = 0b00000000；		/* 提供至外围硬件的时钟频率（fxp） = fx（= 8 MHz）

					  -> 提供至 CPU 的时钟频率（fcpu） = fxp = 8 MHz */

/*------------------------------------------------------------------------------
	初始化端口 0
------------------------------------------------------------------------------*/
	P0   = 0b00000000；	/*设置 P00-P03 的输出锁存为低电平*/
	PM0  = 0b11110000；	/* 将 P00-P03 设置为输出模式 */

/*------------------------------------------------------------------------------
	初始化端口 2
------------------------------------------------------------------------------*/
	P2   = 0b00000000；	/*设置 P20-P23 的输出锁存为低电平 */

	PM2  = 0b11110000；	/* 将 P20-P23 设置为输出模式 */

/*------------------------------------------------------------------------------
	初始化端口 3
------------------------------------------------------------------------------*/
	P3   = 0b00000000；	/* 设置 P30-P33 的输出锁存为低电平 */

	PM3  = 0b11110000；	/* 将 P30-P33 设置为输出模式 */

/*------------------------------------------------------------------------------
	初始化端口 4
------------------------------------------------------------------------------*/
	P4   = 0b00001000；	/* 设置 P40-P42 和 P44-P47 的输出锁存为低电平，P43 的输出锁存为高
电平（串行发送时的设置） */
	PM4  = 0b00010000；	/* 将 P40-P43 和 P45-P47 设置为输出模式，P44 设置为输入模式 */
```

```
/*-------------------------------------------------------------------------
        初始化端口 12
-------------------------------------------------------------------------*/

        P12   = 0b00000000；   /* 设置 P120-P123 的输出锁存为低电平 */

        PM12  = 0b11110000；  /* 将 P120-P123 设置为输出模式 */

/*-------------------------------------------------------------------------

        初始化端口 13

-------------------------------------------------------------------------*/

        P13   = 0b00000001；   /* 设置 P130 的输出锁存为高电平 */

/*-------------------------------------------------------------------------
        设置 UART6
-------------------------------------------------------------------------*/

        CKSR6 = 1；                    /* 将波特率设置为 9600 bps */
        BRGC6 = 208；                  /*（同上） */
        ASIM6 = 0b00011000；           /* 偶校验输出，7 位字符长度， 1 个停止位 */
                                       /* INTSRE6 作为基于错误发生时产生的中断 */
        POWER6 = 1；                   /* 内部工作时钟操作使能 */
        TXE6  =        1；             /* 发送操作使能 */
        RXE6  =        1；             /* 接收操作使能 */

        return；
}

/****************************************************************************

        Main loop

****************************************************************************/
void main（void）
{
        g_ucRxCnt = 0；                /* 接收计数值 = 0 */
        g_ucStoreAddr = 0；            /* 初始化缓冲区的起始写入地址 */
        g_ucReadAddr = 0；             /* 初始化缓冲区的起始读取地址 */
        IF0 = 0x00；                   /* 预先清除无效中断请求 */
        SRMK6 = 0；                    /* INTSR6（串行接收）中断使能 */
        SREMK6 = 0；                   /* INTSRE6（接收错误）中断使能 */
        EI（）；                               /* 中断向量使能 */

        while（1）
        {
                while（g_ucRxCnt == 0）        /* 等待接收中断 */
                {
                        NOP（）；
                }

                while（g_ucRxCnt > 0）         /* 接收计数值> 0 时的处理 */
                {
```

```
g_ucRxData = g_ucRxBuff[g_ucReadAddr]；      /* 读取数据 */
g_ucRxCnt -= 1；                        /*接收计数值减 1 */

if（!g_ucRxData.7）                     /* 正常接收期间的处理 */

{
        switch（g_ucRxData）
        {
                case 'T'：              /*接收 T（54H）时 */

                        fn_uart_send（'O'）；    /* 发送 O（4FH） */

                        fn_uart_send（'K'）；    /* 发送 K（4BH） */

                        fn_uart_send（0x0D）；           /* 发送换行码“CR” */

                        fn_uart_send（0x0A）；           /* 发送换行码“LF” */
                        break；

                case 't'：              /* 接收 t（74H）时 */

                        fn_uart_send（'o'）；    /* 发送 o（6FH） */
                        fn_uart_send（'k'）；    /* 发送 k（6BH） */
                        fn_uart_send（0x0D）；           /* 发送换行码“CR” */

                        fn_uart_send（0x0A）；           /* 发送换行码“LF” */

                        break；

                default ：              /* 接收其它数据时 */

                        fn_uart_send（'U'）；    /* 发送 U（55H） */
                        fn_uart_send（'C'）；    /* 发送 C（43H） */
                        fn_uart_send（0x0D）；           /* 发送换行码“CR” */

                        fn_uart_send（0x0A）；           /* 发送换行码“LF” */

                        break；
        }
}

else                            /* 接收错误时的处理 */
{
        if（g_ucRxData.2）      /* 产生奇偶校验错误时 */

        {
                fn_uart_send（'P'）；    /* 发送 P（50H） */
                fn_uart_send（'E'）；    /* 发送 E（45H） */
                fn_uart_send（0x0D）；           /* 发送换行码“CR” */

                fn_uart_send（0x0A）；           /* 发送换行码“LF” */

        }

        if（g_ucRxData.1）      /* 产生帧错误时 */
```

```
                {
                        fn_uart_send（'F'）；    /* 发送 F（46H） */
                        fn_uart_send（'E'）；    /* 发送 E（45H） */
                        fn_uart_send（0x0D）；           /* 发送换行码“CR” */

                        fn_uart_send（0x0A）；           /* 发送换行码“LF” */

                }

                if（g_ucRxData.0）       /* 产生溢出错误时 */

                {
                        fn_uart_send（'O'）；    /* 发送 O（4FH） */
                        fn_uart_send（'E'）；    /* 发送 E（45H） */
                        fn_uart_send（0x0D）；           /* 发送换行码“CR” */
                        fn_uart_send（0x0A）；           /* 发送换行码“LF” */

                }
            }
            g_ucReadAddr += 1；               /* 读取地址加 1 */

            if（g_ucReadAddr >= BUFF_SIZE）      /* 当读取地址在缓冲区外时 */

            {
                g_ucReadAddr = 0；                /* 初始化缓冲区的起始读取地址 */

            }
        }
    }
}

/*****************************************************************************

    串行接收中断 INTSR6

*****************************************************************************/
__interrupt void fn_intsr6（）
{
    unsigned char ucData；

    ucData = RXB6；                              /* 读取串行接收数据 */

    if（g_ucRxCnt < BUFF_SIZE）  /* 写入地址在缓冲区内时 */

    {
        g_ucRxCnt += 1；                         /* 接收计数值加 1 */

        g_ucRxBuff[g_ucStoreAddr] = ucData；  /* 存储接收数据 */

        g_ucStoreAddr += 1；                     /* 写入地址加 1 */

        if（g_ucStoreAddr >= BUFF_SIZE）      /* 写入地址在缓冲区外时 */

        {
            g_ucStoreAddr = 0；                 /* 初始化缓冲区的起始写入地址 */
```

```
            }
        }

        return；
}

/******************************************************************************

        接收错误中断 INTSRE6

******************************************************************************/
__interrupt void fn_intsre6（）
{
        unsigned char ucData；
        unsigned char ucTemp；

        ucData = ASIS6 | 0b10000000；        /* 设置位 7 的错误标志以存储错误信息 */

        ucTemp = RXB6；                      /* 读取（丢弃）串行接收数据 */

        if（g_ucRxCnt < BUFF_SIZE） /* 写入地址在缓冲区内时 */

        {
                g_ucRxCnt += 1；                            /* 接收计数值加 1 */

                g_ucRxBuff[g_ucStoreAddr] = ucData； /* 存储接收数据 */

                g_ucStoreAddr += 1；                 /* 写入地址加 1 */

                if（g_ucStoreAddr >= BUFF_SIZE）     /* 写入地址在缓冲区外时 */
                {
                        g_ucStoreAddr = 0；          /* 初始化缓冲区的起始写入地址 */

                }
        }

        return；
}

/*==============================================================================
        串行数据发送函数

 该函数用来发送串行数据。
 利用下述函数可发送由“Data”指示的 1 字节数据。

        fn_uart_send（Data）；
==============================================================================*/
void fn_uart_send（unsigned char ucTxData）
{
        g_ucAsif6 = ASIF6；          /* 读取发送状态 */

        while（g_ucAsif6.1）         /* 等待发送使能 */
        {
                g_ucAsif6 = ASIF6；  /* 读取发送状态 */
        }
        TXB6 = ucTxData；                    /* 串行发送 */
```

```
        return;
}
```

● op.asm（汇编语言版和 C 语言版共用）

```
; ================================================================================
;
;       选项字节
;
; ================================================================================
OPBT   CSEG   AT      0080H
     DB     10011000B                    ;  选项字节区。
;              || ||||
;           || |||+------------------------------   内部低速振荡器可由软件停止。
;           || |++-------------------------------   所使用的晶体或陶瓷振荡时钟。
;           || +---------------------------------   P34/RESET 引脚用作 RESET 引脚。
;           ++-----------------------------------   接通电源时或复位解除后的振荡稳定时间 = 2^10/fx。

     DB     11111111B ;                  保护字节区（用于自编程模式）。
;           ||||||||
;           ++++++++-------------------------   所有模块都能写入或擦除。

end
```

附录B　版本修订历史

版本	发布日期	页码	修订
第一版	2008年02月	–	–

详细信息请联系：

中国区

MCU 技术支持热线：
电话：+86-400-700-0606 (普通话)
服务时间：9:00-12:00，13:00-17:00 （不含法定节假日）

网址：
http://www.cn.necel.com/ （中文）
http://www.necel.com/ （英文）

[北京]

日电电子（中国）有限公司
中国北京市海淀区知春路 27 号
量子芯座 7，8，9，15 层
电话：（+86）10-8235-1155
传真：（+86）10-8235-7679

[上海]

日电电子（中国）有限公司上海分公司
中国上海市浦东新区银城中路 200 号
中银大厦 2409-2412 和 2509-2510 室
电话：（+86）21-5888-5400
传真：（+86）21-5888-5230

上海恩益禧电子国际贸易有限公司
中国上海市浦东新区银城中路 200 号
中银大厦 2511-2512 室
电话：（+86）21-5888-5400
传真：（+86）21-5888-5230

[深圳]

日电电子（中国）有限公司深圳分公司
深圳市福田区益田路卓越时代广场大厦 39 楼
3901，3902，3909 室
电话：（+86）755-8282-9800
传真：（+86）755-8282-9899

[香港]

香港日电电子有限公司
香港九龙旺角太子道西 193 号新世纪广场
第 2 座 16 楼 1601-1613 室
电话： （+852）2886-9318
传真： （+852）2886-9022
2886-9044

[成都]

日电电子（中国）有限公司成都分公司
成都市二环路南三段 15 号天华大厦 7 楼 703 室
电话：(+86)28-8512-5224
传真：(+86)28-8512-5334